TIE-02306

Introduction to Software Engineering

5 credit units

08-QAtest-ItSE2019-v3

Course contents (plan)

1. Course basics, intro
2. Sw Eng in general, overview
3. Requirements
4. Different software systems
5. Basic UML Diagrams (”Class”, Use Case, Navigation)
6. Life Cycle models
7. UML diagrams, in more detail
8. **Quality and Testing**
9. Project work
10. Project management
11. Open source, APIs, IPR
12. Embedded systems
13. Recap

8. Quality and Testing

- GSD = global sw dev, distributed sw dev, outsourcing, off-shoring
- maintenance
- Quality Assurance (QA)
- testing
  - review / inspection
  - unit testing
  - integration testing
  - system testing
  - usability testing
  - acceptance testing
- TDD (Test-Driven Development)
- version control
- configuration management
- verification and validation (V&V)
-

Current at course (w 44)

- **WE7 this week** (development processes)
- after those we think next week WE8 groups

- **continue updating your Trello (kanban) boards** = use at your process

- **EXAM 1/3 (28 students, 4..35 min, 7..12 points)**
- ***remember EXAM 2/3 (weeks 44-46); diagrams (Dia or EA tool)***

- **1st presentations group-to-group feedback at PRP system (link via Moodle)**

Tietotekniikan yö 5 (5th ICT Night), 05.11.2019, TB104, 17-
IT-Hekuma 13.11.2019 at Tietotalo and Sähkötalo.
InnoEvent (04-08.11.2019) www.innoevent.fi

Some annual DAYs…

International Programmer's Day 13.09.2019

Software Freedom Day 21.09.2019

International Project Management Day 07.11.2019

World Usability Day 14.11.2019

World Quality Day 14.11.2019

Leftovers from previous lecture set…

- GSD = global sw development; development teams in different countries, ideally 24 hours development in a day by different time-zones, BUT a lot of risks (culture, process, management,…)
- distributed sw development; development teams are not located in the same place
- **outsourcing**; some other company does the work (FI: ulkoistaminen)
- **subcontracting**; some other company does (part of) the work (FI: alihankinta)
- **off-shoring**; work is done at another country
  - main reasons for off-shoring; get cheap talented work (but does it happen ?)
- **reshoring**; getting work back home country from abroad
- main reasons for reshoring; delivery time, total cost, quality.

The main problem: is the Sw Dev process understood the same way everywhere ? There are very few success stories.

Why outsource software development?

1 Access to a larger talent pool and the latest technology

2 Increased Focus on Core Business

3 Cost savings

4 Better Risk Management

5 Accommodate peak loads

6 Better Security

7 Spend Less Time on Support

8 Reduce time to market.

[https://hackernoon.com/8-reasons-why-outsourcing-software-development-works-f399fcb5d2d2]

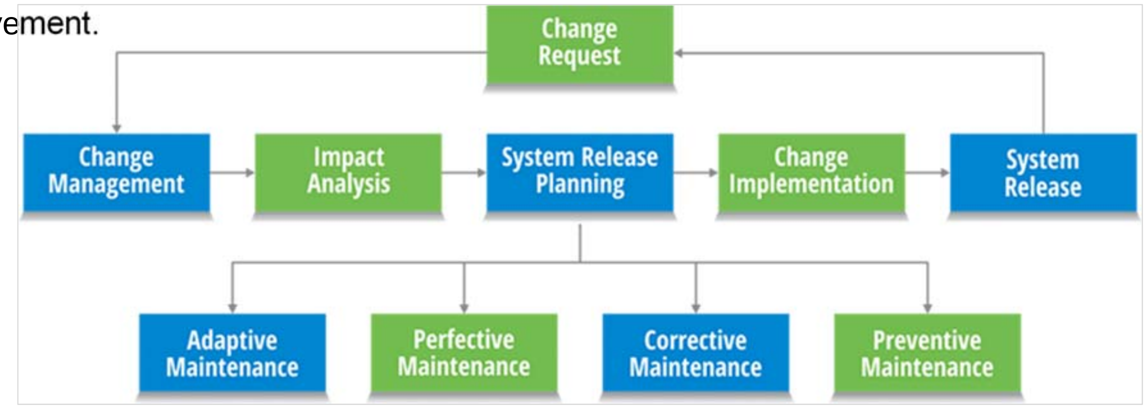
Some outsourcing success stories:
http://www.invimatic.com/how-to-select-a-software-development-outsourcing-company/
https://www.daxx.com/blog/development-trends/outsourcing-success-stories
https://medium.com/@xipetechnology/software-development-outsourcing-success-stories-c0972476245c

Maintenance

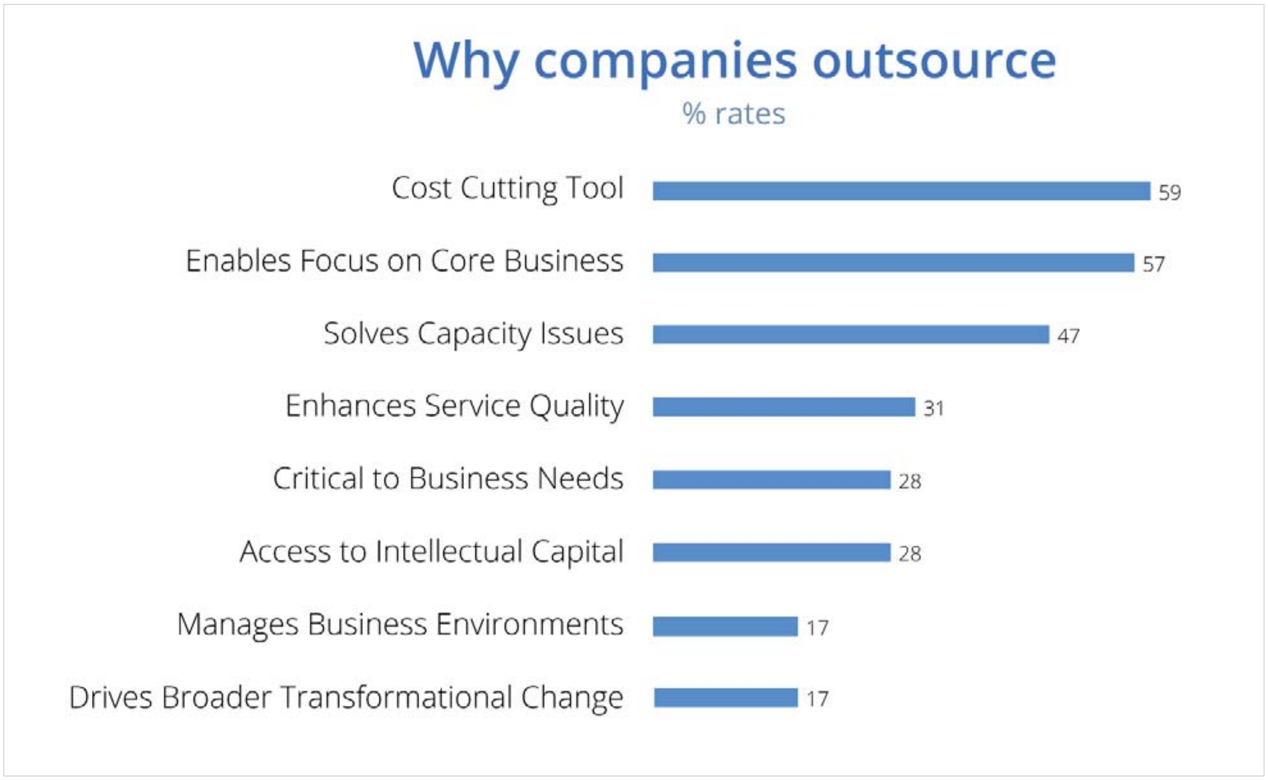
- maintenance = make modifications to code for its extended life; add, change, delete.

Why maintenance

- bug fixing
- capability enhancement
- removal of outdated functrions
- performance improvement.



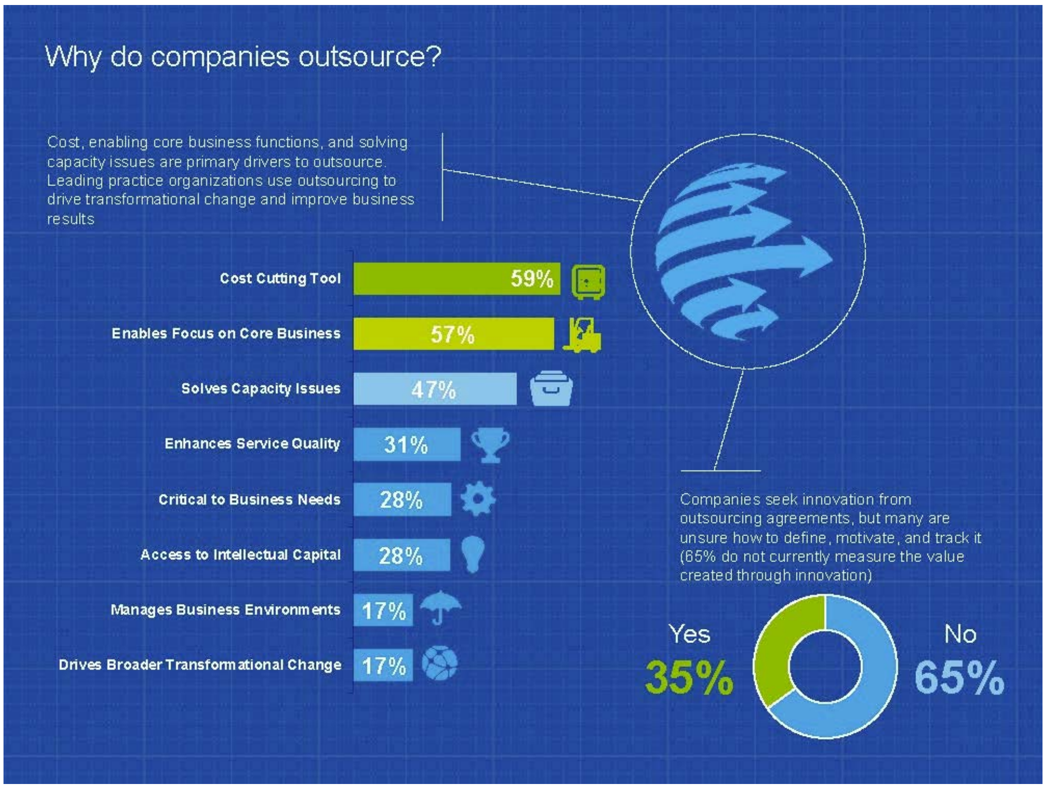
[https://simplified-it-outsourcing.com/blog/why-software-maintenance-is-necessary]



[https://perfectial.com/blog/outsourcing-software-development/]

Tampereen yliopisto
Tampere University

Tampereen yliopisto
Tampere University



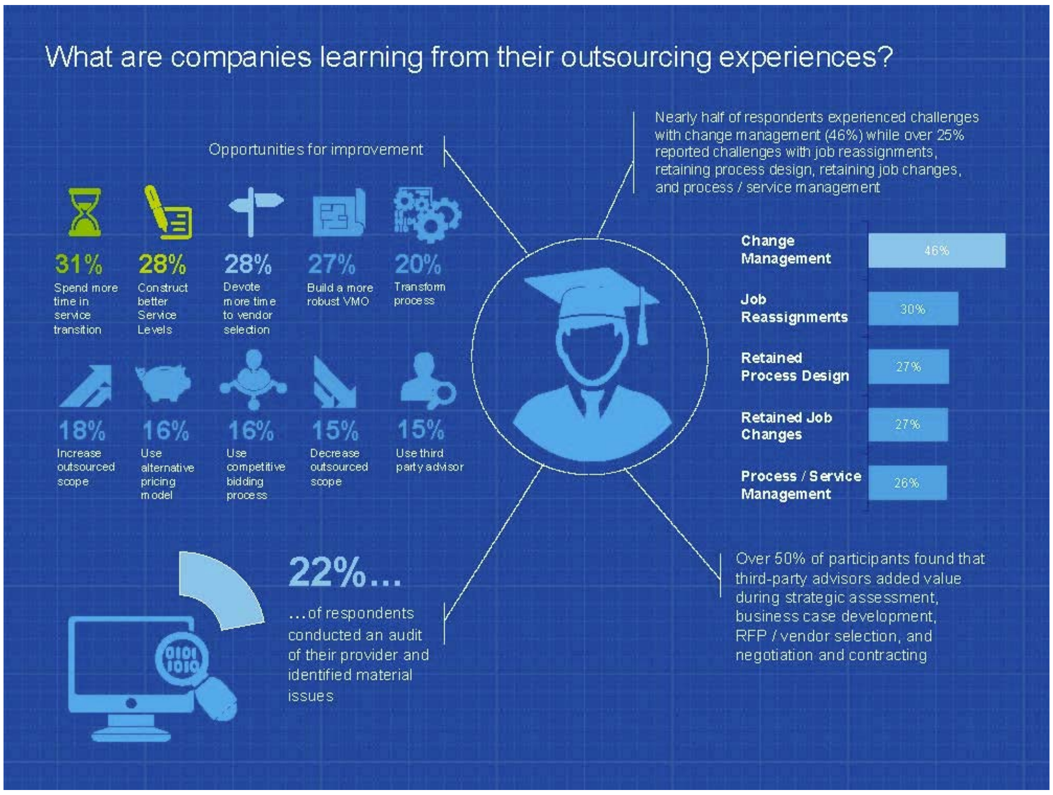
Why do companies outsource?

Cost, enabling core business functions, and solving capacity issues are primary drivers to outsource. Leading practice organizations use outsourcing to drive transformational change and improve business results

Companies seek innovation from outsourcing agreements, but many are unsure how to define, motivate, and track it (65% do not currently measure the value created through innovation)

Cost Cutting Tool 59%
Enables Focus on Core Business 57%
Solves Capacity Issues 47%
Enhances Service Quality 31%
Critical to Business Needs 28%
Access to Intellectual Capital 28%
Manages Business Environments 17%
Drives Broader Transformational Change 17%
Yes 35%
No 65%

[www.deloitte.com , 2016]



What are companies learning from their outsourcing experiences?

Opportunities for improvement

- 31% Spend more time in service transition
- 28% Construct better Service Levels
- 28% Devote more time to vendor selection
- 27% Build a more robust VMO
- 20% Transform process
- 18% Increase outsourced scope
- 16% Use alternative pricing model
- 16% Use competitive bidding process
- 15% Decrease outsourced scope
- 15% Use third party advisor

Nearly half of respondents experienced challenges with change management (46%) while over 25% reported challenges with job reassignments, retaining process design, retaining job changes, and process / service management

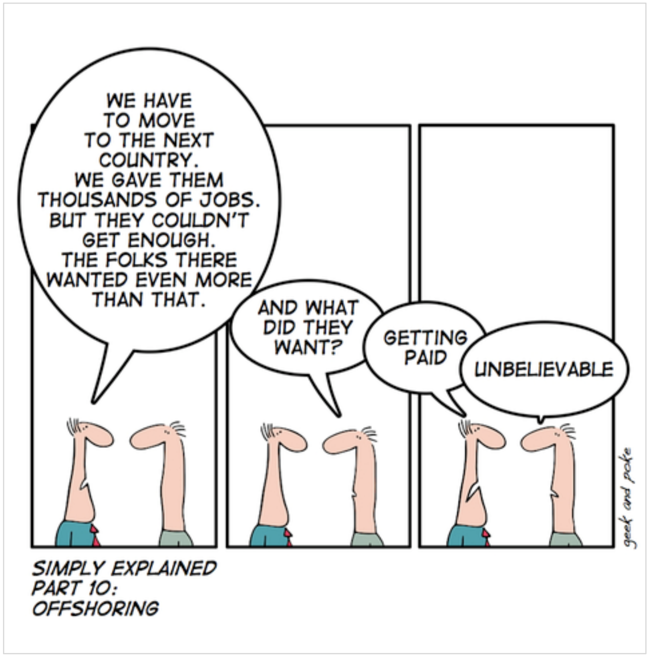
Change Management	46%
Job Reassignments	30%
Retained Process Design	27%
Retained Job Changes	27%
Process / Service Management	26%

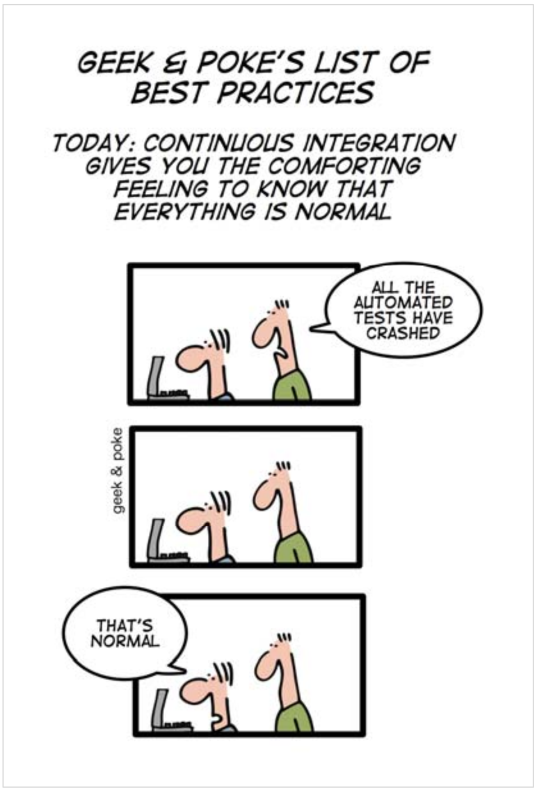
22%...

...of respondents conducted an audit of their provider and identified material issues

Over 50% of participants found that third-party advisors added value during strategic assessment, business case development, RFP / vendor selection, and negotiation and contracting

[www.deloitte.com , 2016]


WE HAVE TO MOVE TO THE NEXT COUNTRY. WE GAVE THEM THOUSANDS OF JOBS. BUT THEY COULDN'T GET ENOUGH. THE FOLKS THERE WANTED EVEN MORE THAN THAT.
AND WHAT DID THEY WANT?
GETTING PAID
UNBELIEVABLE
geek and poke
SIMPLY EXPLAINED PART 10: OFFSHORING


GEEK & POKE'S LIST OF BEST PRACTICES
TODAY: CONTINUOUS INTEGRATION GIVES YOU THE COMFORTING FEELING TO KNOW THAT EVERYTHING IS NORMAL
ALL THE AUTOMATED TESTS HAVE CRASHED
geek & poke
THAT'S NORMAL

Where to outsource ?

10 Best Countries to Outsource Software Development, Based on Data
by Dianna Gunn / Updated: September 24, 2019 /
[www.codeinwp.com/blog/best-countries-to-outsource-software-development/]

1. India
2. Ukraine
3. China
4. Poland
5. The Philippines
6. Romania
7. Brazil
8. Taiwan
9. Egypt
10. Canada.

Well… a Finnish consultant said some years ago, that South Africa will be the next "hot spot", after salaries in Estonia, Russia, India and China have raised too much.

3.1240

distributed computing

1. spreading of computation and data across a number of computers connected by a network.

[ISO 24765:2017]

ISO 24765:2017 Systems and software engineering — Vocabulary

3.2314 , maintainability

1. **ease with which a software system or component can be modified to change or add capabilities, correct faults or defects, improve performance or other attributes, or adapt to a changed environment**

2. ease with which a hardware system or component can be retained in, or restored to, a state in which it can perform its required functions

3. capability of the software product to be modified

4. average effort required to locate and fix a software failure

5. speed and ease with which a program can be corrected or changed

6. degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers

cf. extendability, flexibility

Note 1 to entry: **Maintainability** includes installation of updates and upgrades. **Modifications** include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications. Modifications include those carried out by specialized support staff, and those carried out by business or operational staff, or end users.

QA = quality assurance, ongoing process

3.3260 , quality assurance (QA)

1. part of quality management focused on providing confidence that quality requirements will be fulfilled

2. **all the planned and systematic activities implemented within the quality system**, and demonstrated as needed, to provide adequate confidence that an entity will fulfill requirements for quality

Note 1 to entry: [ISO 9000:2005] There are both internal and external purposes for quality assurance: within an organization, quality assurance provides confidence to management; in contractual situations, quality assurance provides confidence to the customer or others. Some quality control and quality assurance actions are interrelated. Unless requirements for quality fully reflect the needs of the user, quality assurance does not necessarily provide adequate confidence.

3.3265 , quality control (QC)

1. set of activities designed to **evaluate the quality** of developed or manufactured products

2. monitoring service performance or product quality, recording results, and recommending necessary changes

cf. quality assurance

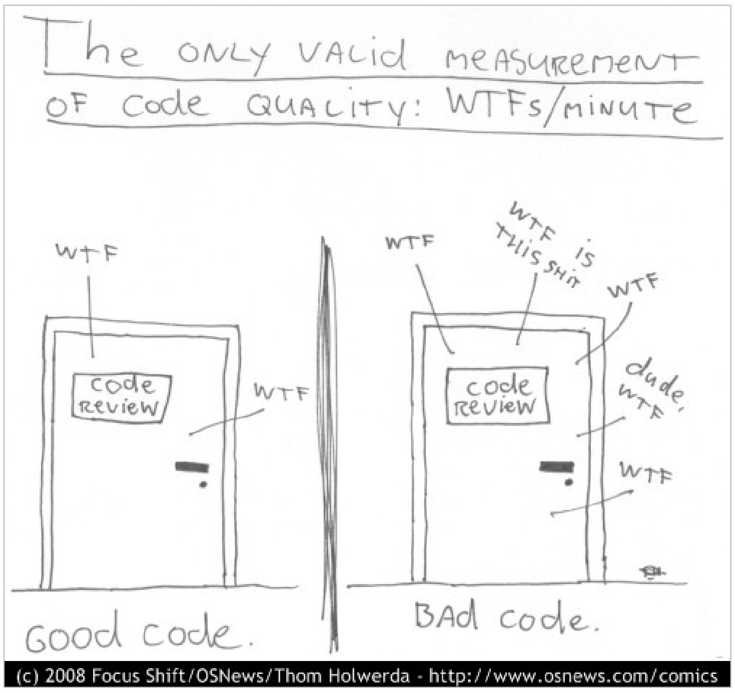
Note 1 to entry: This term has no standardized meaning in software engineering at this time.

[ISO/IEC/IEEE 24765:2017]

Inspections and reviews

Inspection = formal, all material is scanned, log/minutes/diary is kept.

Review = informal, perhaps only main parts of deliverable is scanned, perhaps no any log just author's own memo.


The ONLY VALid MEASUREMENT OF Code QUALITY: WTFs/minute
WTF
Code Review
WTF
WTF
WTF is THIS SHIT
WTF
Code Review
dude, WTF
WTF
Good code.
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

3.2000 , inspection

1. **visual examination** of a software product **to detect and identify** software anomalies, including **errors and deviations** from standards and specifications

2. a static analysis technique that relies on visual examination of development products to detect errors, violations of development standards, and other problems***3.*** examining or measuring to verify whether an activity,

component, product, result, or service conforms to specified requirements

cf. static testing

Note 1 to entry: Inspections are peer examinations led by impartial facilitators who are trained in inspection techniques. Determination of remedial or investigative action for an anomaly is a mandatory element of a software inspection, although the solution could be determined outside the inspection meeting. Types include code inspection and design inspection.

3.3503 , review

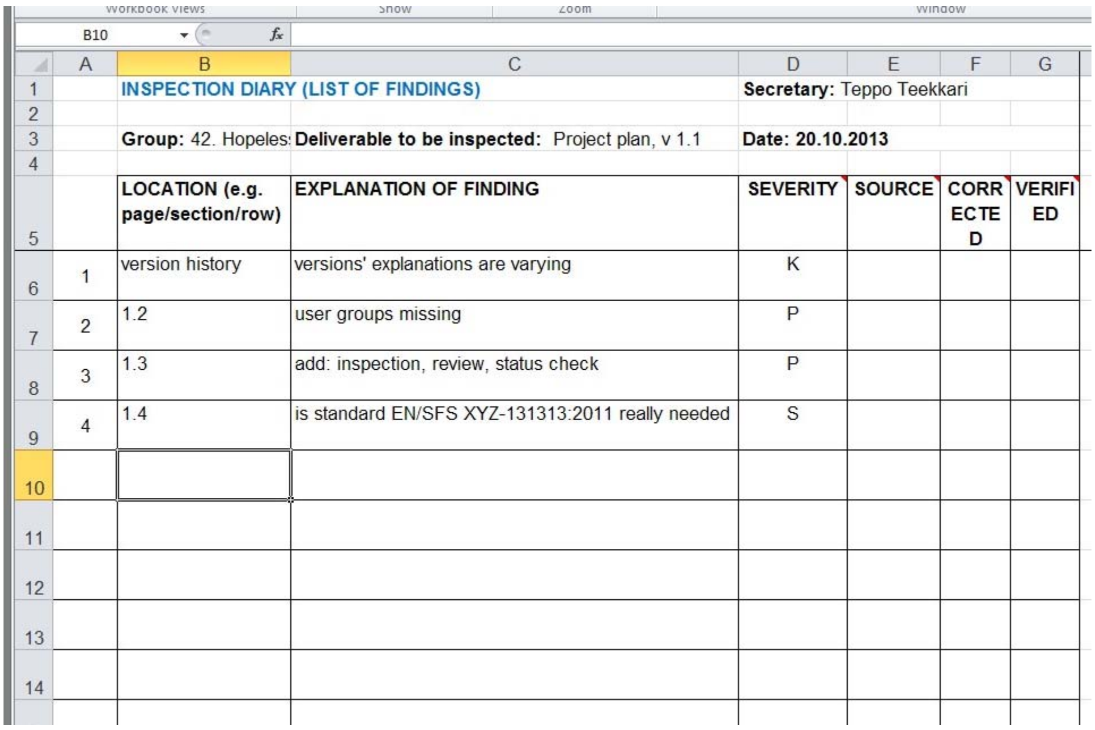
1. **process**, which can include a meeting, in which work products are presented to some stakeholders **for comment or approval**

2. process or meeting during which a software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval

3. process or meeting during which a work product, or set of work products, is presented

to project personnel, managers, users, customers, or other interested parties for comment or approval.

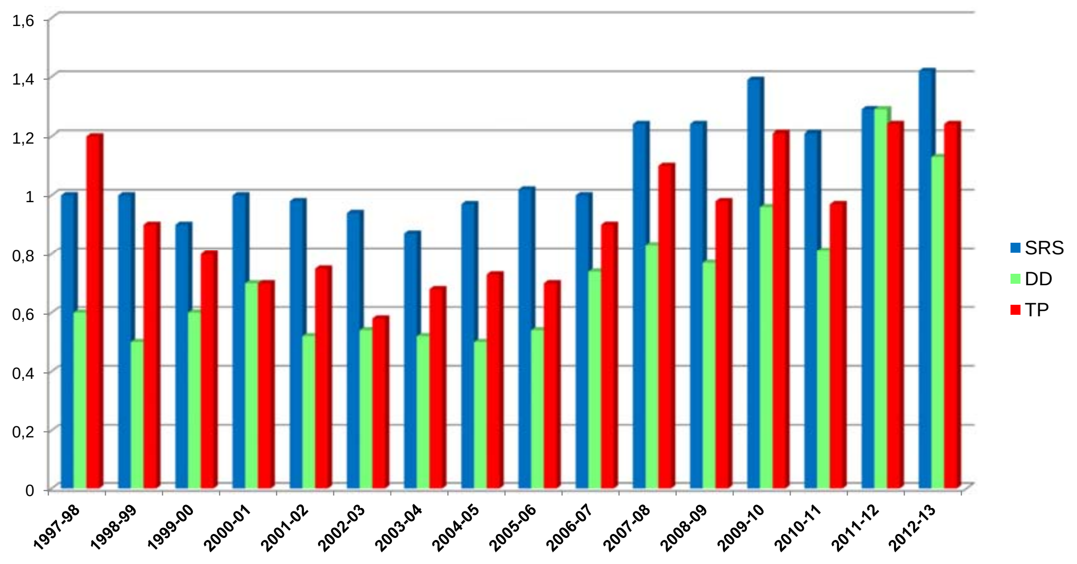
[ISO 24765-2017]



	A	B	C	D	E	F	G
1		INSPECTION DIARY (LIST OF FINDINGS)		**Secretary:** Teppo Teekkari			
2							
3		**Group:** 42. Hopeles	**Deliverable to be inspected:** Project plan, v 1.1	**Date: 20.10.2013**			
4							
5		**LOCATION (e.g. page/section/row)**	**EXPLANATION OF FINDING**	**SEVERITY**	**SOURCE**	**CORR ECTE D**	**VERIFI ED**
6	1	version history	versions' explanations are varying	K			
7	2	1.2	user groups missing	P			
8	3	1.3	add: inspection, review, status check	P			
9	4	1.4	is standard EN/SFS XYZ-131313:2011 really needed	S			
10							
11							
12							
13							
14							

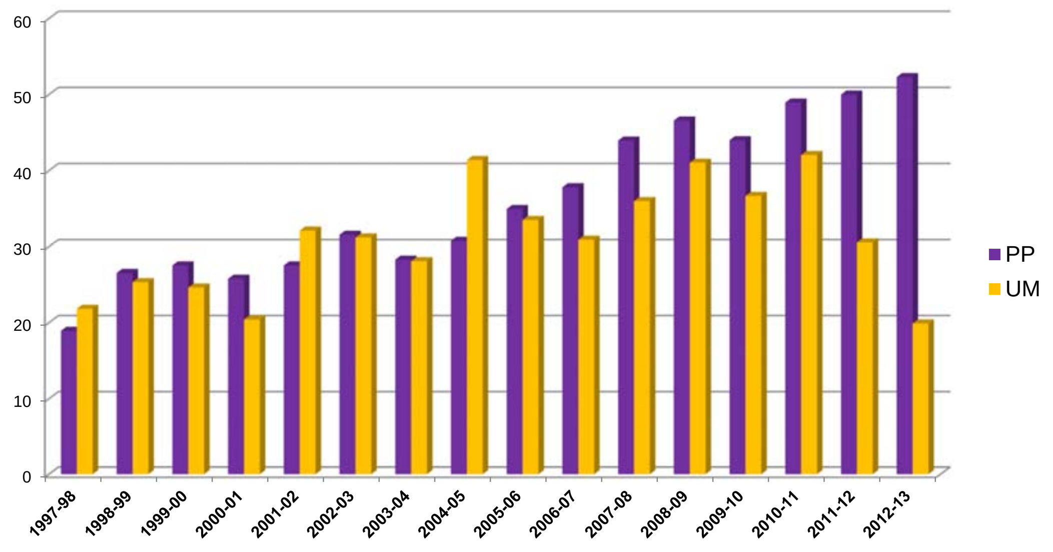
Why to test… what to test…

Document inspections at TUT findings / page (average) (5/11)



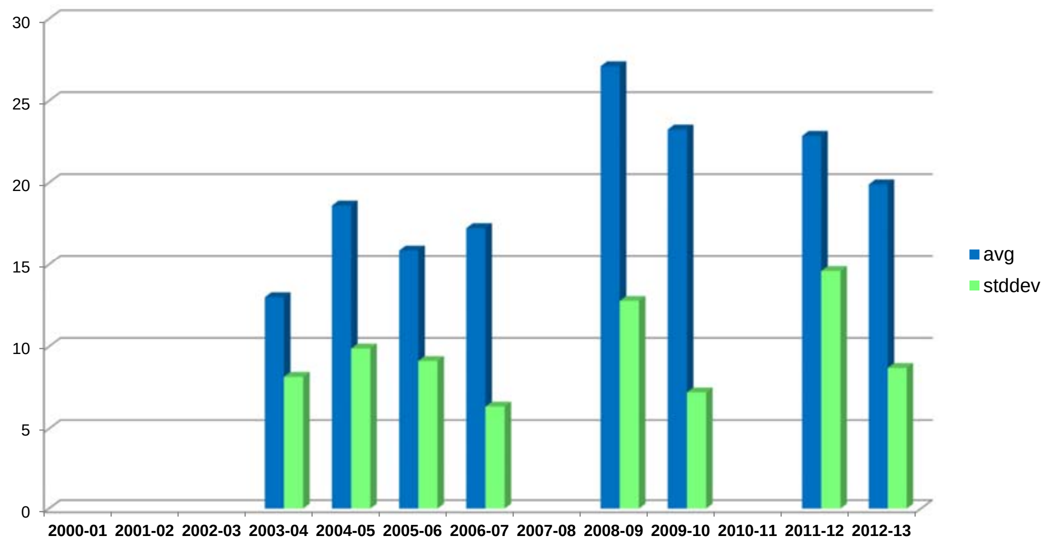
SRS= Sw Requirements Spec.
DD = Design Document
TP = Test Plan

Document inspections at TUT findings (average) (3/11)



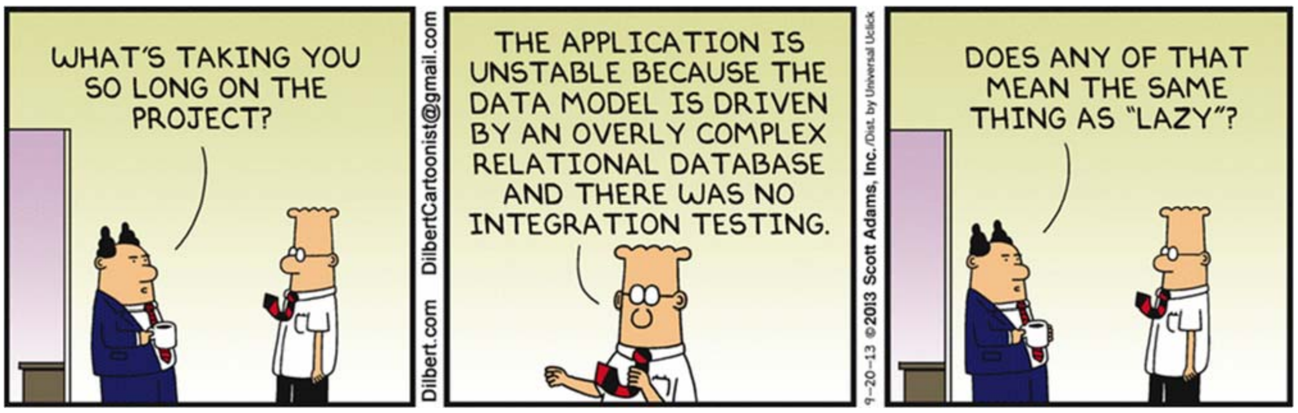
PP = Project Plan
UM = User Manual

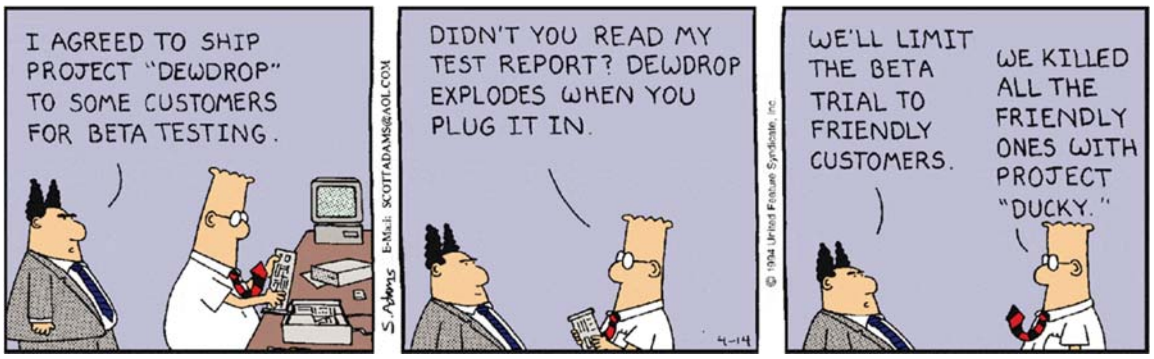
Code inspections at TUT findings (average) (7/11)



Motivation for inspections

- ~30 times more effective in defect finding than testing
- Early defect detection -> “right the first time”
- Spreads information to several people -> reduces risks in losing a key employee
- Effective in training new members in a group
- Results in a better common understanding of the product
- Cases report improved quality in defects, deliveries and schedules
- Manageability and productivity tend to improve as well as employee satisfaction
- Better commitment from the team to the project
- The most easiest and effective way of QA
- Inspections DO affect to product quality
  (but they should be done well) !

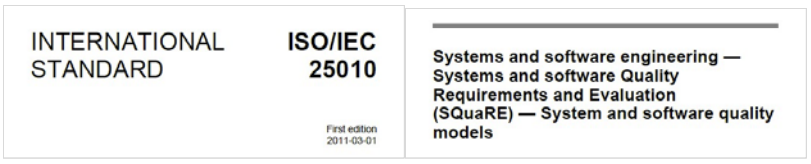

WHAT'S TAKING YOU SO LONG ON THE PROJECT?
THE APPLICATION IS UNSTABLE BECAUSE THE DATA MODEL IS DRIVEN BY AN OVERLY COMPLEX RELATIONAL DATABASE AND THERE WAS NO INTEGRATION TESTING.
DOES ANY OF THAT MEAN THE SAME THING AS "LAZY"?
Dilbert.com DilbertCartoonist@gmail.com
9-20-13 ©2013 Scott Adams, Inc./Dist. by Universal Uclick


I AGREED TO SHIP PROJECT "DEWDROP" TO SOME CUSTOMERS FOR BETA TESTING.
S. Adams E-Mail: SCOTTADAMS@AOL.COM
DIDN'T YOU READ MY TEST REPORT? DEWDROP EXPLODES WHEN YOU PLUG IT IN.
4-14
© 1994 United Feature Syndicate, Inc.
WE'LL LIMIT THE BETA TRIAL TO FRIENDLY CUSTOMERS.
WE KILLED ALL THE FRIENDLY ONES WITH PROJECT "DUCKY."

What is ”quality software” ?

- easy for end-users to understand its use
- easy to maintain (e.g. add funtionality later)
- good code comments
- good architecture (e.g. easy to add components later)
- nice GUI (graphical user interface)
- end-user can to complex functions with a few mouse clicks
- 5000 users at a same time (web application)
- cheap
- delivered in (or even ahead of) time
- it crashes or jams only once in a work-day ?

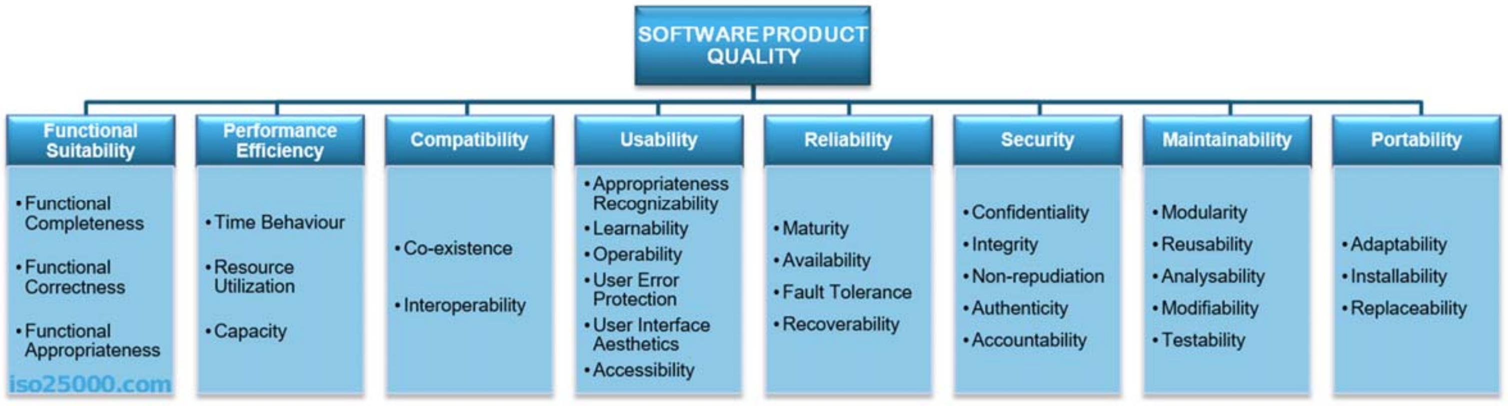
Well… difficult to describe. It depends on the stakeholder from who you ask.


INTERNATIONAL STANDARD

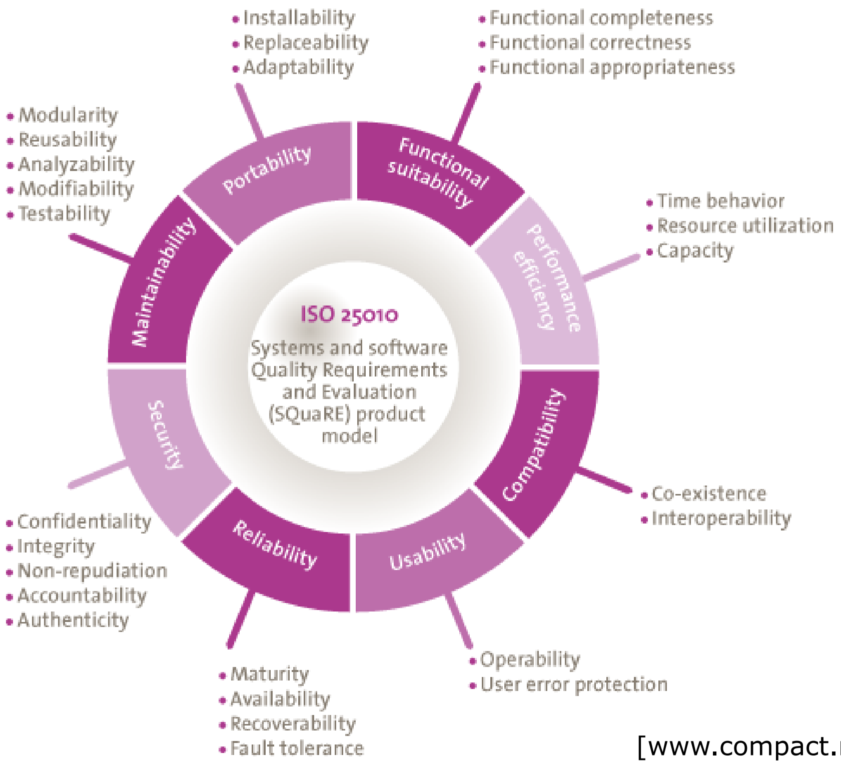
ISO/IEC 25010

First edition 2011-03-01

Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models

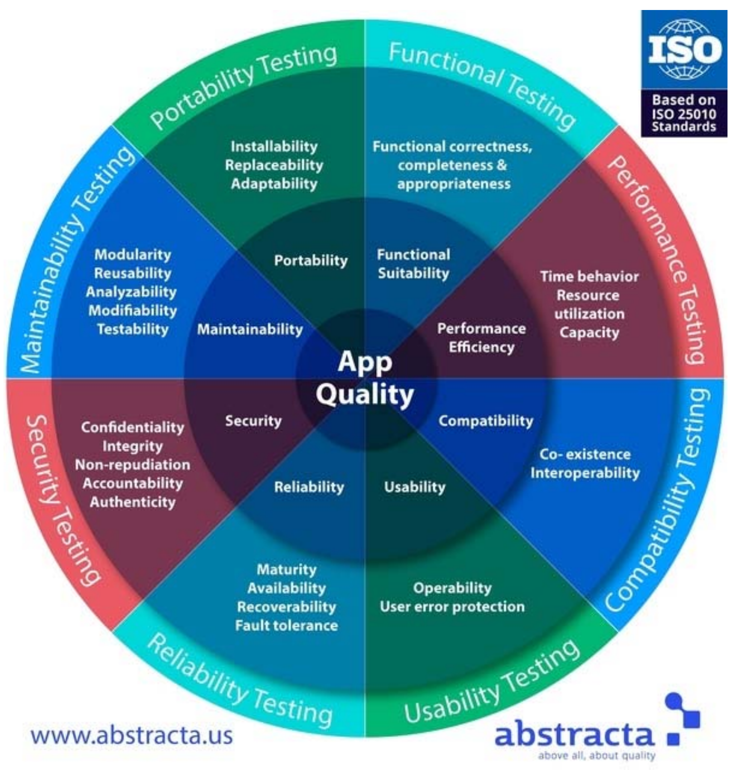


There are many classifications of quality factors

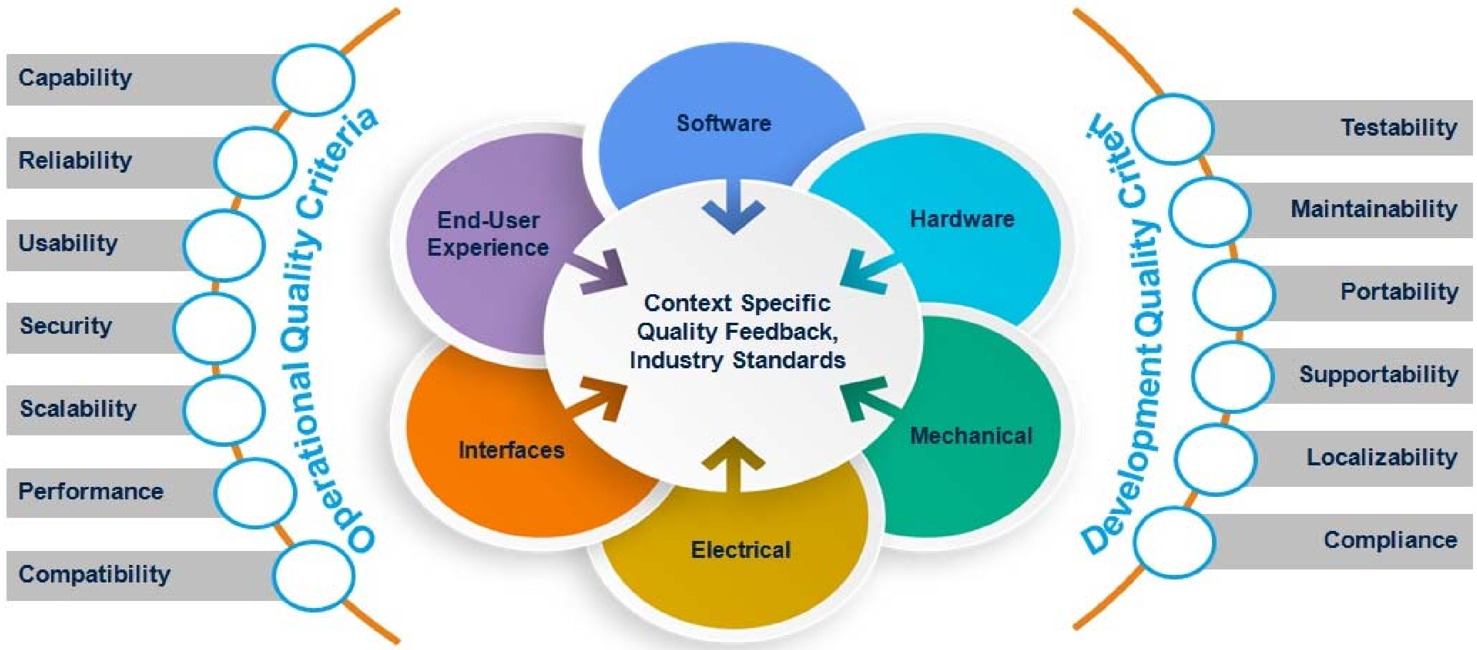


[www.compact.nl]

There are many classifications of quality factors



There are many classifications of quality factors



[Capgemini]

Software quality attributes [Sommerville, 2014]

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Quality is as customer and / or end-user feels it… do never forget them.

Sw quality factors

McCall's Factor Model

- This model classifies all software requirements into 11 software quality factors. The 11 factors are grouped into three categories – product operation, product revision, and product transition factors.
- **Product operation factors** − Correctness, Reliability, Efficiency, Integrity, Usability.
- **Product revision factors** − Maintainability, Flexibility, Testability.
- **Product transition factors** − Portability, Reusability, Interoperability.

[Tutorialspoint]

ISO 25000 standards set

Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE)

- ISO/IEC 25000 - *Guide to SQuaRE*
- ISO/IEC 25001 - *Planning and Management*
- ISO/IEC 25010 - *System and software quality models*
- ISO/IEC 25012 - *Data Quality model*
- ISO/IEC 25020 - *Measurement reference model and guide*
- ISO/IEC 25021 - *Quality measure elements*
- ISO/IEC 25022 - *Measurement of quality in use*
- ISO/IEC 25023 - *Measurement of system and software product quality*
- ISO/IEC 25024 - *Measurement of data quality*
- ISO/IEC 25030 - *Quality requirements*
- ISO/IEC 25040 - *Evaluation reference model and guide*
- ISO/IEC 25041 - *Evaluation guide for developers, acquirers and independent evaluators*
- ISO/IEC 25042 - *Evaluation module*
- ISO/IEC 25045 - *Evaluation module for recoverability.*

ISO 10000-standards set

- ISO 10001:2007, *Quality management — Customer satisfaction — Guidelines for codes of conduct for organizations*
- ISO 10002:2014, *Quality management — Customer satisfaction — Guidelines for complaints handling in organizations*
- ISO 10003:2007, *Quality management — Customer satisfaction — Guidelines for dispute resolution external to organizations*
- ISO 10004:2012, *Quality management — Customer satisfaction — Guidelines for monitoring and measuring*
- ISO 10005:2005, *Quality management systems — Guidelines for quality plans*
- ISO 10006:2003, *Quality management systems — Guidelines for quality management in projects*
- ISO 10007:2003, *Quality management systems — Guidelines for configuration management*
- ISO 10008, *Quality management — Customer satisfaction — Guidelines for business-to-consumer electronic commerce transactions.*

You can not add ”quality” to a product. Quality must be present from the very beginning in the process.

3.3259 , quality

- ***1.*** degree to which the system **satisfies** the stated and implied **needs** of its various stakeholders, and thus provides **value**
- ***2.*** ability of a product, service, system, component, or process to meet customer or user **needs**, expectations, or requirements
- ***3.*** the degree to which a set of inherent characteristics fulfils **requirements**.

[ISO 24765:2017]

QA , QC

3.3260 , quality assurance (QA)

- ***1.*** part of quality management focused on **providing confidence** that quality requirements will be fulfilled
- ***2.*** all the planned and systematic activities implemented within the quality system, and demonstrated as needed, to provide adequate confidence that an entity will fulfill requirements for quality.

3.3265 , quality control (QC)

- ***1.*** set of activities designed to **evaluate the quality** of developed or manufactured products
- ***2.*** monitoring service performance or product quality, recording results, and recommending necessary changes.

[ISO 24765:2017]

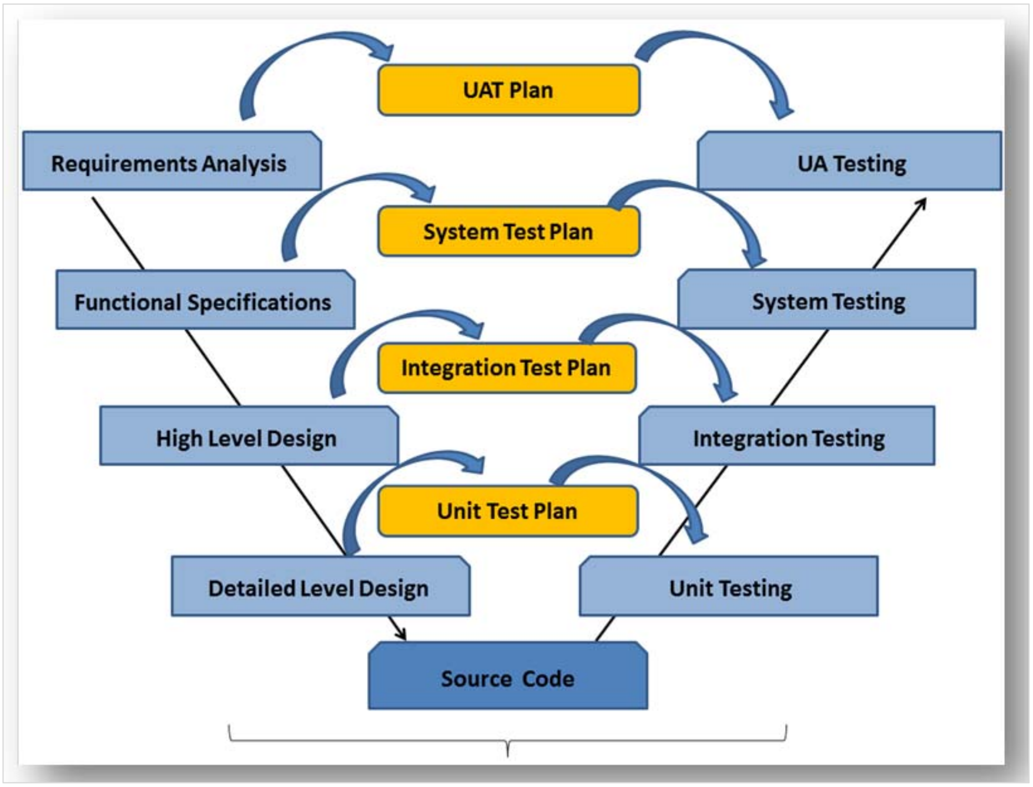
SQA , SQC

- **Software Quality Assurance** – Software Quality Assurance (SQA) is a set of activities to **ensure the quality in software engineering processes** that ultimately result in quality software products. The activities establish and evaluate the processes that produce products. It involves process-focused action.

- **Software Quality Control** – Software Quality Control (SQC) is a set of activities to **ensure the quality in software products**. These activities focus on determining the defects in the actual products produced. It involves product-focused action.

[Tutorialspoint]

Testing phases



[qainsights.com/]

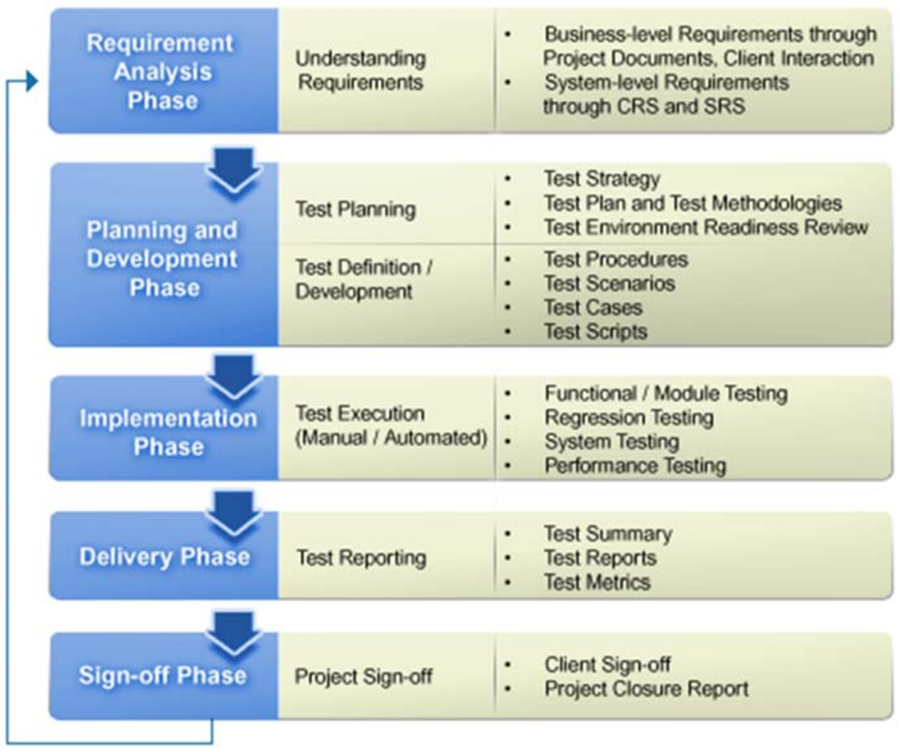
different kind of testing

Remember also reviews and inspections, before testing.

- **unit testing**, earlier module testing (one component)
- **smoke testing** (is it stable enough for system testing)
- **integration testing**, several modules/components are tested together
- **system testing** (the whole product)
- **reggression testing** (run old tests again, and compare to old results)
- **usability testing** (by actual end-users, or specialists test), e.g. **exploratory** testing (FI: tutkiva testaus)
- special testing; **security** testing, **stress** testing (e.g. server load),…
- **UAT = user acceptance testing**
- **acceptence testing** (the final delivered system).

Use some **issue tracker** or **bug database** to keep track on errors and corrections.

Testing in SDLC phases

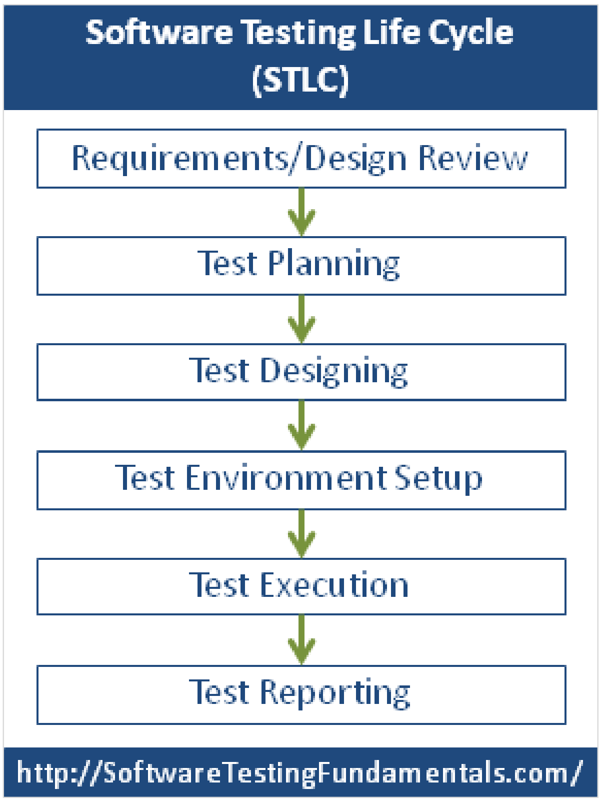


Requirement Analysis Phase	Understanding Requirements	• Business-level Requirements through Project Documents, Client Interaction • System-level Requirements through CRS and SRS
Planning and Development Phase	Test Planning	• Test Strategy • Test Plan and Test Methodologies • Test Environment Readiness Review
	Test Definition / Development	• Test Procedures • Test Scenarios • Test Cases • Test Scripts
Implementation Phase	Test Execution (Manual / Automated)	• Functional / Module Testing • Regression Testing • System Testing • Performance Testing
Delivery Phase	Test Reporting	• Test Summary • Test Reports • Test Metrics
Sign-off Phase	Project Sign-off	• Client Sign-off • Project Closure Report

[www.360logica.com/test-management/test-models-planning/]

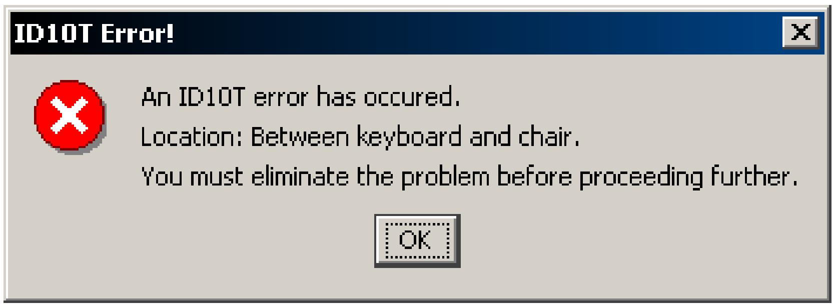
You have to PLAN your test, and also have some LOG or REPORT to prove you have tested something.
There must be some evidence of testing.

Just to say ”we did exploratory testing two hours, and did not found any bugs” is NOT enough.



Software Testing Life Cycle (STLC)
Requirements/Design Review
Test Planning
Test Designing
Test Environment Setup
Test Execution
Test Reporting
http://SoftwareTestingFundamentals.com/

ID-10-T error (FI: ID sata TTI -virhe)


ID10T Error!

An ID10T error has occured.
Location: Between keyboard and chair.
You must eliminate the problem before proceeding further.

This is NOT a test plan: ”We will do some system testing, if we have time.”

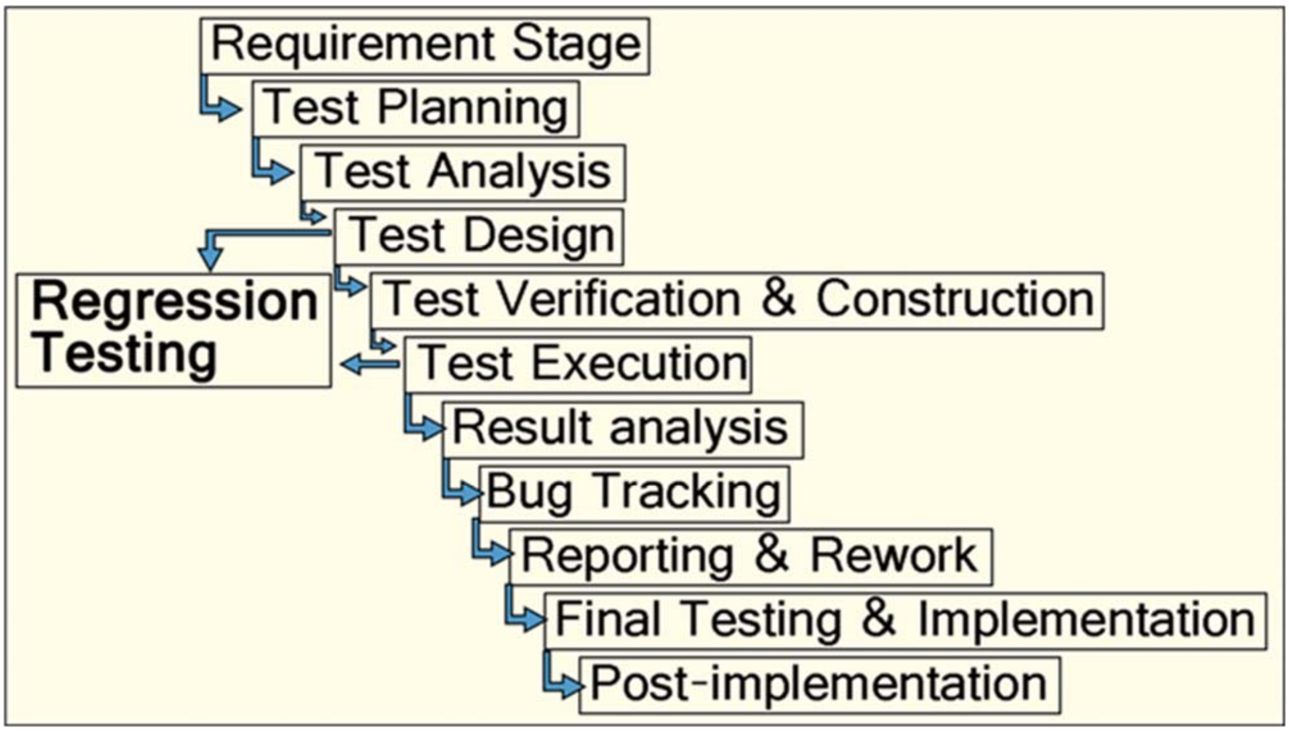
Regression testing

✧ Regression testing is testing the system **to check that changes have not ‘broken’ previously working code**.

✧ In a manual testing process, regression testing is expensive but, with automated testing, it is simple and straightforward. All tests are rerun every time a change is made to the program.

✧ Tests must run ‘successfully’ before the change is committed.

To check there have not been any “side-effects” after modifications.



[techspirited.com/software-testing-life-cycle]

Types of user testing

✧ **Alpha testing**

- Users of the software work with the development team to test the software **at the developer's site**.

✧ **Beta testing**

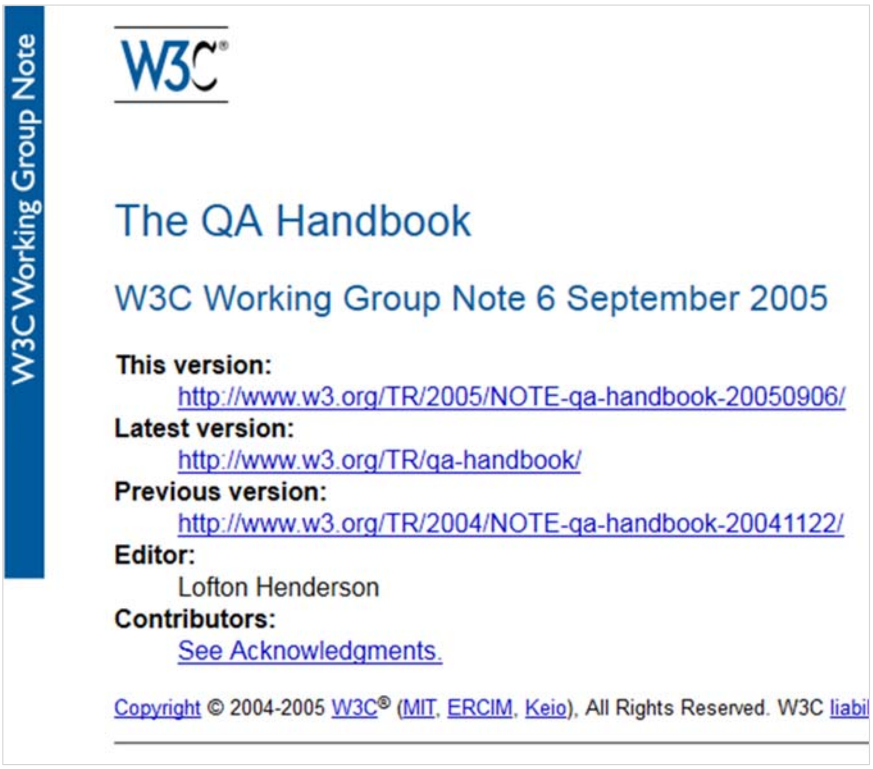
- A release of the software is **made available to users** to allow them to experiment and to raise problems that they discover with the system developers.

✧ **Acceptance testing**

- **Customers test a system** to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment. Primarily for custom systems.

Tampereen yliopisto
Tampere University

The QA Handbook (QAH) is a non-normative handbook about the process and operational aspects of certain quality assurance practices of W3C's Working Groups, with particular focus on testability and test topics. It is intended for Working Group chairs and team contacts. It aims to help them to **avoid known pitfalls and benefit from experiences** gathered from the W3C Working Groups themselves. It provides **techniques, tools, and templates** that should facilitate and accelerate their work.



The QA Handbook

W3C Working Group Note 6 September 2005

This version:
http://www.w3.org/TR/2005/NOTE-qa-handbook-20050906/
Latest version:
http://www.w3.org/TR/qa-handbook/
Previous version:
http://www.w3.org/TR/2004/NOTE-qa-handbook-20041122/
Editor:
Lofton Henderson
Contributors:
See Acknowledgments.

Copyright © 2004-2005 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liabi

Error.. or what… terminology varies

Error is a deviation from the actual and the expected result. It represents the mistakes made by the people.

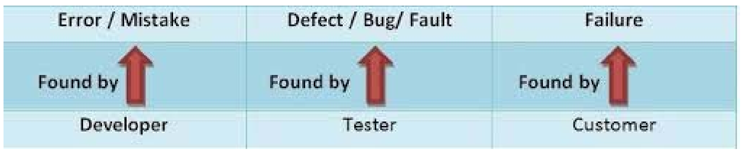
Faults are the result of an error. It is the incorrect step or process due to which the program or the software behaves in an unintended manner

Bug is an evidence of Fault in a program due to which program does not behave in an intended manner

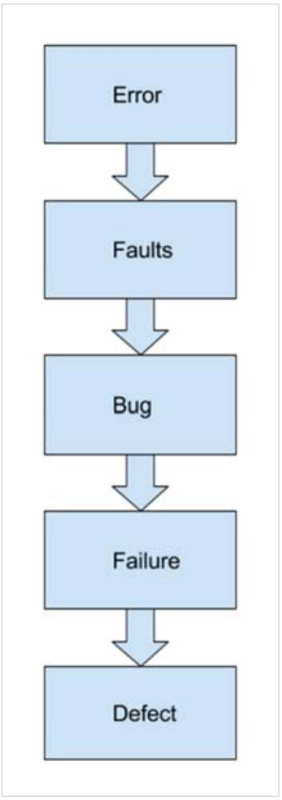
Failure is an inability of the system or components to perform its required function. Failure occurs when Faults executes

Defect is said to be detected when Failure occurs.

[stackoverflow.com/]



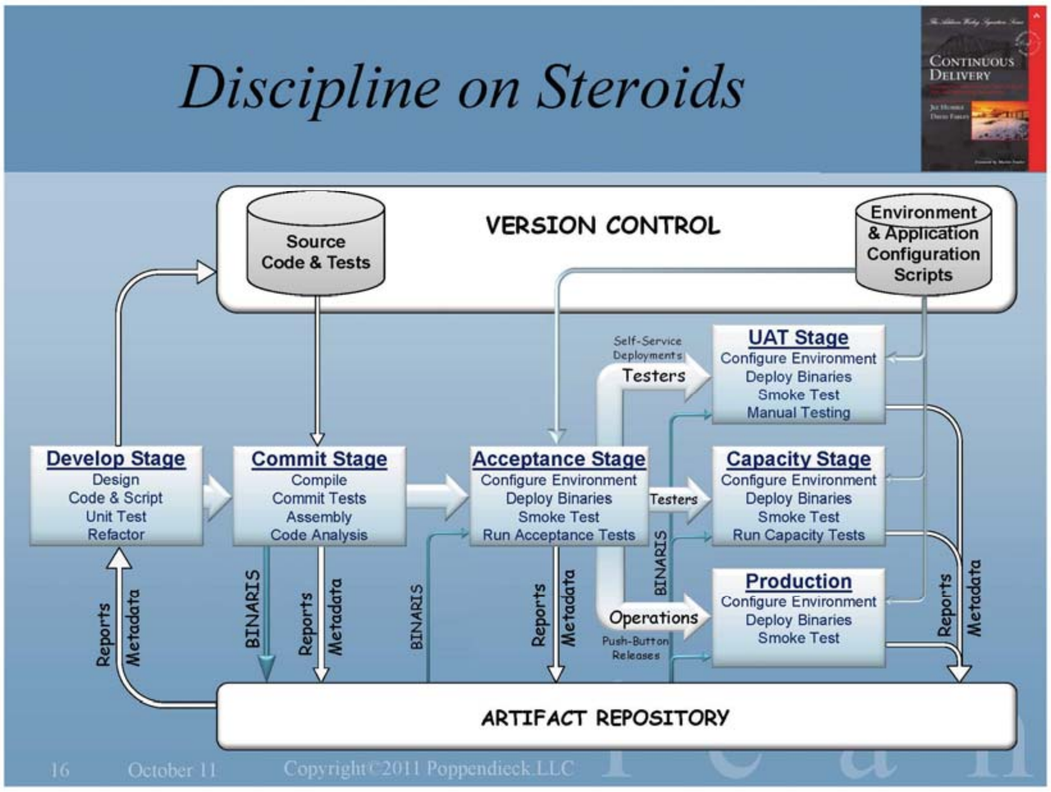
Error / Mistake	Defect / Bug/ Fault	Failure
Found by	Found by	Found by
Developer	Tester	Customer



CI = continuous integration

CD = continuous delivery

UAT = user acceptance testing

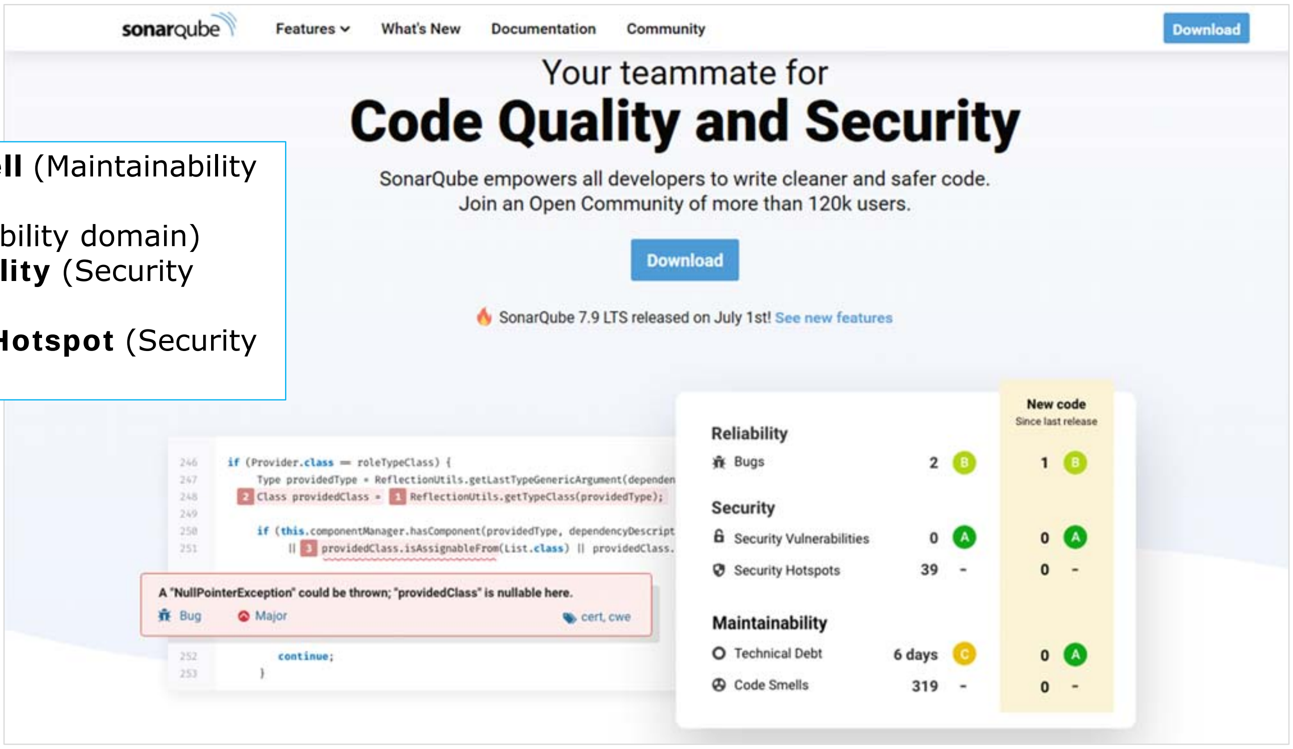


SonarQube tool, some measurements

Project, e.g. ”Sonar Groovy” or ”sonar-test” (in GitHub)

- **Home** (**bugs&vulnerabilities**, code smells, coverage, duplications)
- **Issues**
  - **Type** (bug, vulnerability, **code smell**)
  - **Resolution** (**unresolved**, fixed, false positive, won’t fix, removed)
- **Measures**
  - **Reliability** (bugs.. rating … effort)
  - **Security** (vulnerabilities … rating … effort)
  - **Maintainability** (code smells … rating … **technical debt**)
  - **Coverage** (line … condition … uncovered … unit tests … duration)
  - **Duplications** (blocks … lines … files)
  - **Size** (LOC … lines … statemenmts … functions … classes … files … directories … comments)
  - **Complexity** (function … file … class)
  - **Issues**
- **Code** (common numbers shortly)
- **Activity** (by date).

- **Code Smell** (Maintainability domain)
- **Bug** (Reliability domain)
- **Vulnerability** (Security domain)
- **Security Hotspot** (Security domain)



Your teammate for

Code Quality and Security

SonarQube empowers all developers to write cleaner and safer code.
Join an Open Community of more than 120k users.

SonarQube 7.9 LTS released on July 1st! See new features

```
if (Provider.class == roleTypeClass) {
    Type providedType = ReflectionUtils.getLastTypeGenericArgument(dependen
  2 Class providedClass = 1 ReflectionUtils.getTypeClass(providedType);

    if (this.componentManager.hasComponent(providedType, dependencyDescript
        || 3 providedClass.isAssignableFrom(List.class) || providedClass.
```

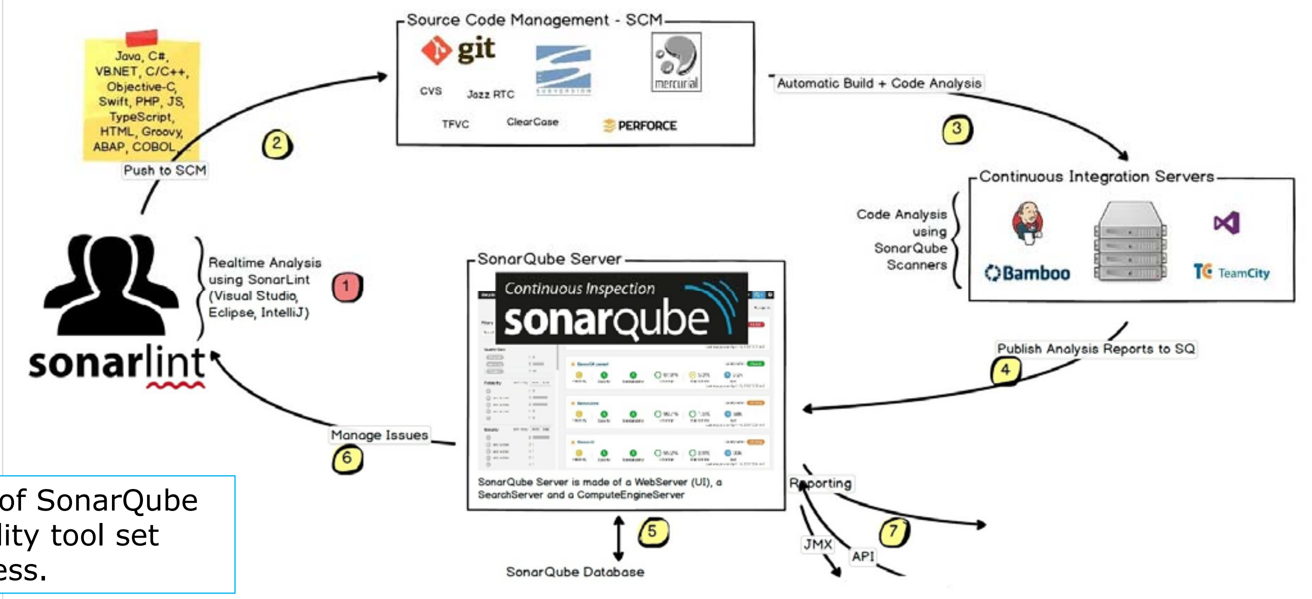
A "NullPointerException" could be thrown; "providedClass" is nullable here.
Bug Major cert, cwe

```
        continue;
    }
```

			New code Since last release	
Reliability				
Bugs	2	B	1	B
Security				
Security Vulnerabilities	0	A	0	A
Security Hotspots	39	-	0	-
Maintainability				
Technical Debt	6 days	C	0	A
Code Smells	319	-	0	-

[www.sonarqube.org/]

Tampereen yliopisto
Tampere University



Source Code Management - SCM
git
CVS
Jazz RTC
TFVC
ClearCase
PERFORCE
mercurial
Java, C#, VB.NET, C/C++, Objective-C, Swift, PHP, JS, TypeScript, HTML, Groovy, ABAP, COBOL
Push to SCM
2
Automatic Build + Code Analysis
3
Continuous Integration Servers
Code Analysis using SonarQube Scanners
Bamboo
TeamCity
Realtime Analysis using SonarLint (Visual Studio, Eclipse, IntelliJ)
1
sonarlint
SonarQube Server
Continuous Inspection
sonarqube
SonarQube Server is made of a WebServer (UI), a SearchServer and a ComputeEngineServer
Publish Analysis Reports to SQ
4
Manage Issues
6
5
SonarQube Database
Reporting
JMX
API
7

Example of SonarQube code quality tool set and process.

https://blog.codinghorror.com/code-smells/

- Developing your "code nose" is something that happens early in your programming career, if it's going to happen at all.
- Combined code; most of these smells should be familiar to you.

Code Smells Within Classes

- **Comments**
- **Long Method**
- **Long Parameter List**
- **Duplicated code**
- **Conditional Complexity**
- **Combinitorial Explosion**
- **Large Class**
- **Type Embedded in Name**
- **Uncommunicative Name (method)**
- **Inconsistent Names**
- **Dead Code**
- **Speculative Generality**
- **Oddball Solution**
- **Temporary Field.**

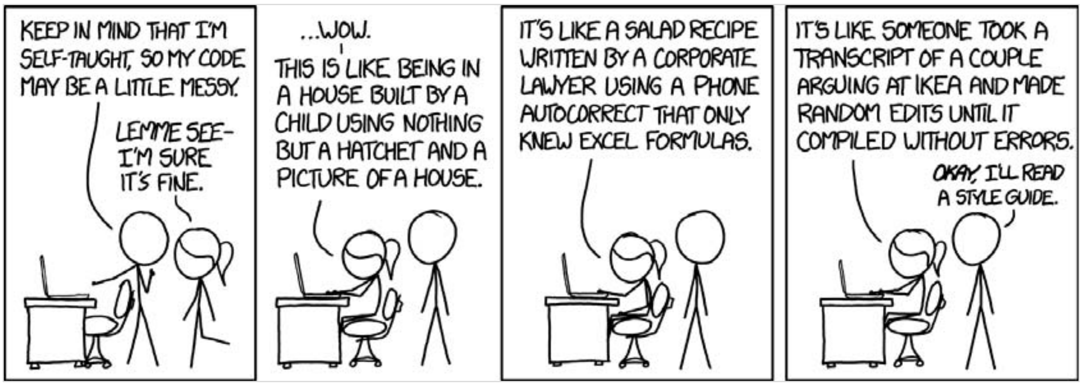
Mika Mäntylä: code smell

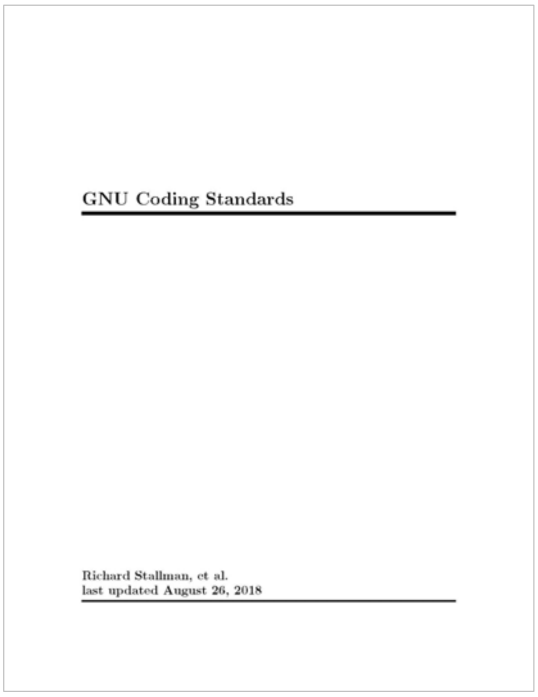
Group name **(Smells in group)**

- **The Bloaters** **(**Long Method, Large Class, Primitive Obsession, Long Parameter List, DataClumps**)**
- **The Object-Orientation Abusers** **(**Switch Statements, Temporary Field, Refused Bequest, Alternative Classes with Different Interfaces**)**
- **The Change Preventers** **(**Divergent Change, Shotgun Surgery, Parallel Inheritance Hierarchies **)**
- **The Dispensables** **(**Lazy class, Data class, Duplicate Code, Dead Code, Speculative Generality**)**
- **The Couplers** **(**Feature Envy, Inappropriate Intimacy, Message Chains, Middle Man**).**

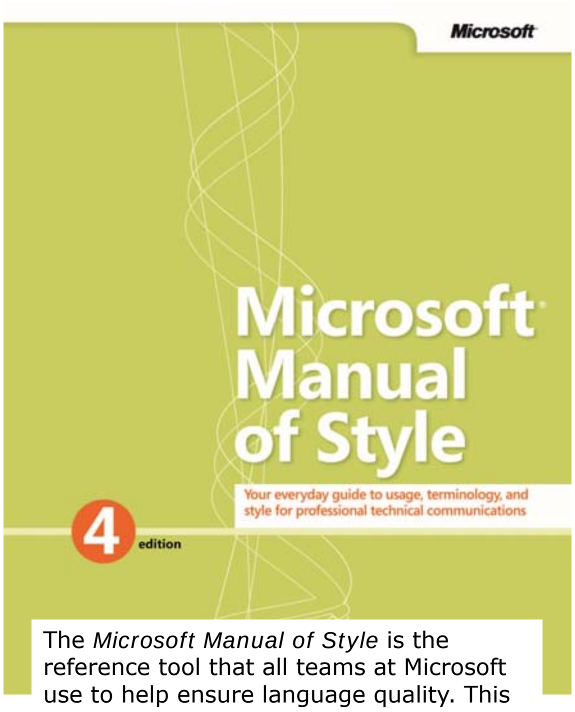
Some Coding Conventions / Style Guides

- **Google C++ Style Guide**
- https://google.github.io/styleguide/cppguide.html
- **Google Java Style Guide**
- https://google.github.io/styleguide/javaguide.html
- **Mozilla codebase coding style**
- https://developer.mozilla.org/en-US/docs/Mozilla/Developer_guide/Coding_Style


KEEP IN MIND THAT I'M SELF-TAUGHT, SO MY CODE MAY BE A LITTLE MESSY.
LEMME SEE- I'M SURE IT'S FINE.
...WOW.
THIS IS LIKE BEING IN A HOUSE BUILT BY A CHILD USING NOTHING BUT A HATCHET AND A PICTURE OF A HOUSE.
IT'S LIKE A SALAD RECIPE WRITTEN BY A CORPORATE LAWYER USING A PHONE AUTOCORRECT THAT ONLY KNEW EXCEL FORMULAS.
IT'S LIKE SOMEONE TOOK A TRANSCRIPT OF A COUPLE ARGUING AT IKEA AND MADE RANDOM EDITS UNTIL IT COMPILED WITHOUT ERRORS.
OKAY, I'LL READ A STYLE GUIDE.


GNU Coding Standards

Richard Stallman, et al.
last updated August 26, 2018


Microsoft
Microsoft Manual of Style
Your everyday guide to usage, terminology, and style for professional technical communications
4 edition

The *Microsoft Manual of Style* is the reference tool that all teams at Microsoft use to help ensure language quality. This guide lays the foundation for the language in our products and services

CI / CD

- Continuous Integration is a software development practice where members of a team **integrate their work frequently**, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.

[Martin Fowler, 2006]

CI

Continuous Integration is the practice of **merging code changes into a shared repository several times a day in order to release a product version at any moment**. This requires an integration procedure which is reproducible and automated.

[Agile Alliance]

continuous integration

Teams practicing **continuous integration** seek two objectives:

- minimize the duration and effort required by each integration episode
- be able to deliver a product version suitable for release at any moment
- In practice, this dual objective requires an integration procedure which is reproducible at the very least, and largely automated. This is achieved through version control tools, team policies and conventions, and tools specifically designed to help achieve continuous integration.

[Agile Alliance]

CD

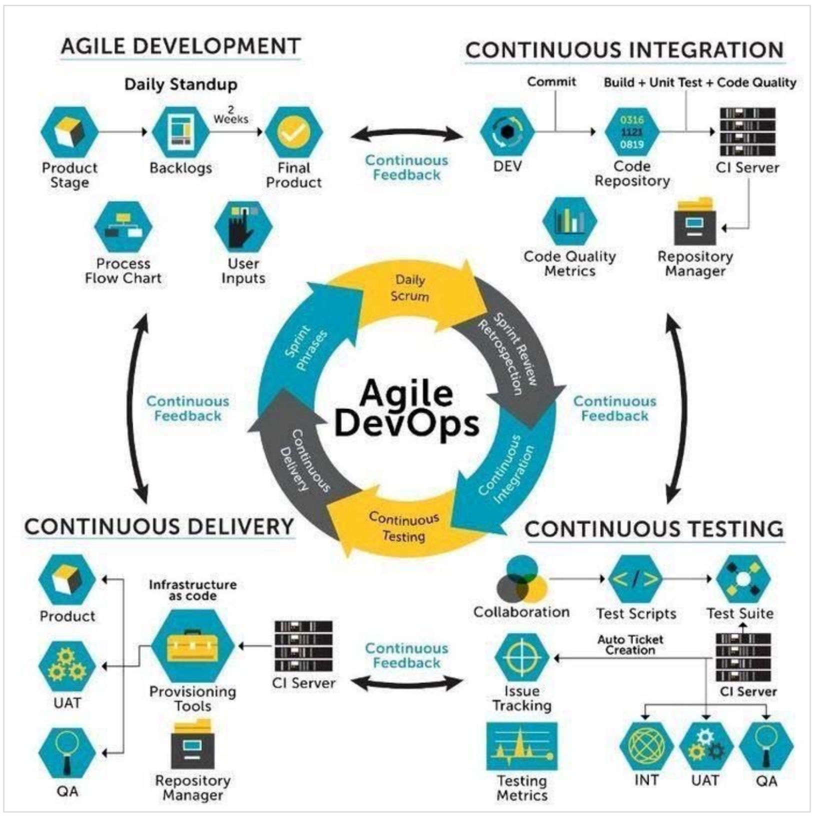
Continuous deployment aims to **reduce the time elapsed between writing a line of code and making that code available to users in production**. To achieve continuous deployment, the team relies on infrastructure that automates and instruments the various steps leading up to deployment, so that after each integration successfully meeting these release criteria, the live application is updated with new code.

[Agile Alliance]

Continuous deployment

- Continuous deployment can be thought of as an extension of continuous integration, aiming at minimizing lead time, the time elapsed between development writing one new line of code and this new code being used by live users, in production.
- To achieve continuous deployment, the team relies on infrastructure that automates and instruments the various steps leading up to deployment, so that after each integration successfully meeting these release criteria, the live application is updated with new code.
- Instrumentation is needed to ensure that any suggestion of lowered quality results in aborting the deployment process, or rolling back the new features, and triggers human intervention.

[Agile Alliance]



automated testing

[Jukka Ala-Mutka]

What are the qualities of a good software bug report?

Anyone can write a bug report. But not everyone can write a effective bug report. You should be able to distinguish between average bug report and a good bug report. How to distinguish a good or bad bug report? It’s simple, apply following characteristics and techniques to report a bug.

1) Having clearly specified bug number:
2) Reproducible:
3) Be Specific:

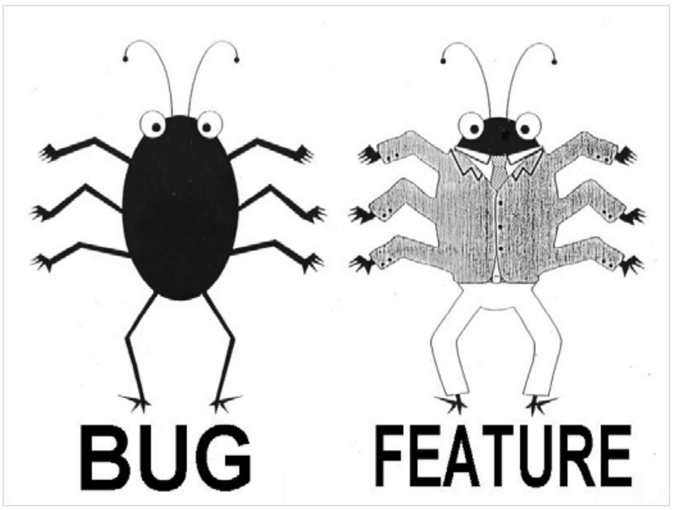
http://www.softwaretestinghelp.com/

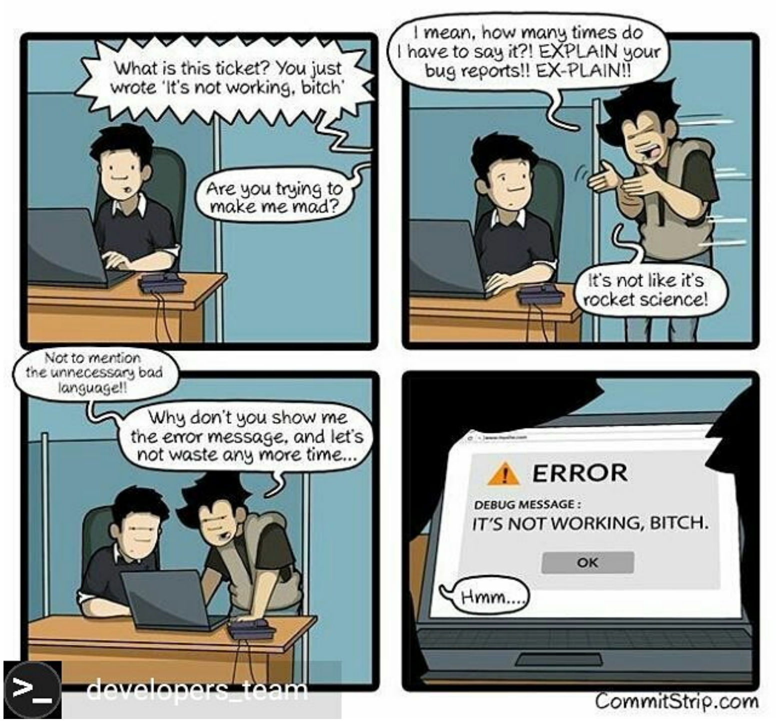
How to Report a Bug?

Use following simple Bug report template:

- **Reporter:** Your name and email address.
- **Product:** In which product you found this bug.
- **Version:** The product version if any.
- **Component:** These are the major sub modules of the product.
- **Platform:** Mention the hardware platform where you found this bug.
- **Operating system:** Mention all operating systems where you found the bug.
- **Priority:** When bug should be fixed? Priority is generally set from P1 to P5, P1 as highest.
- **Severity:** This describes the impact of the bug.
  Types of Severity:
  - **Blocker:** No further testing work can be done.
  - **Critical:** Application crash, Loss of data.
  - **Major:** Major loss of function.
  - **Minor:** minor loss of function.
  - **Trivial:** Some UI enhancements.
  - **Enhancement:** Request for new feature or some enhancement in existing one.
- **Status:**
- **Assign To:**
- **URL:**
- **Summary:**
- **Description:**

http://www.softwaretestinghelp.com/




What is this ticket? You just wrote 'It's not working, bitch'
Are you trying to make me mad?
I mean, how many times do I have to say it?! EXPLAIN your bug reports!! EX-PLAIN!!
It's not like it's rocket science!
Not to mention the unnecessary bad language!!
Why don't you show me the error message, and let's not waste any more time...
ERROR
DEBUG MESSAGE :
IT'S NOT WORKING, BITCH.
OK
Hmm....
developers_team
CommitStrip.com

Bug Bounty

hackr.fi Bug Bounty, **CROWDSOURCED SECURITY TESTING**
Hackrfi enables and manages bug bounty programs. With us you'll create a useful bug bounty program and get great support through its life cycle.

hackerone.com Internet Bug Bounty, **The Most Trusted Hacker-Powered Security Platform**

bugcrowd.com **PUBLIC BUG BOUNTY LIST**
The most comprehensive, up to date crowdsourced list of bug bounty and security disclosure programs from across the web curated by the hacker community.

Test-driven development (TDD)

- ✧ **Test-driven development** (**TDD**) is an approach to program development in which you inter-leave testing and code development.
- ✧ **Tests are written before code** and ‘passing’ the tests is the critical driver of development.
- ✧ You develop code incrementally, along with a test for that increment. You don’t move on to the next increment until the code that you have developed passes its test.
- ✧ TDD was introduced as part of agile methods such as Extreme Programming. However, it can also be used in plan-driven development processes.

Versioning terminology… it depends…

Version (terminology varies…)

Revision (nothing at ISO 24765 std; as you see, terminology varies…)

- "In software development a **revision** is seen as a **major release** of the software that will introduce new features and functionality to the application. In the engineering world revisions are done to show changes to a design so that the changes are documented for the entire design team and any other individual who uses the data."
- "A new **revision** contains minor changes, usually only corrections. A new **version** has major changes, which may include additions and reorganization of the contents."

Release (terminology varies…)

Build "a build is usually a version of software in pre-release format that is used only by the software development company".

Build… an internal version

3.431 , build

1. operational version of a system or component that incorporates a specified subset of the capabilities that the final product will provide

2. process of generating (archiving) **an executable and testable system** from source versions or baselines

3. to perform the steps required to produce **an instance of the product**

Note 1 to entry: In software, this means processing source files to derive target files. In hardware, this means assembling a physical object. The build needs to compile and link the various versions in the correct order. The build tools can be integrated into a configuration management tool.

[ISO 24765:2017]

Release… particular version deployed

3.3382 , release

1. **particular version of a configuration item that is made available for a specific purpose** *[IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering*, 2.12*; ISO/IEC 90003:2014 Software engineering — Guidelines for the application of ISO 9001:2008 to computer software*, 3.10*]*

2. collection of new or changed configuration items that are tested and introduced into a live environment together *[IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering*, 2.1*; ISO/IEC 19770-1:2012 Information technology — Software asset management — Part 1: Processes and tiered assessment of conformance*, 3.12*]*

3. collection of one or more new or changed configuration items deployed into the live environment as a result of one or more changes *[ISO/IEC 19770-5:2015 Information technology — IT asset management — Part 5: Overview and vocabulary*, 3.28*]*

4. software version that is made formally available to a wider community *[IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering*, 2.1*]*

5. Delivered version of an application which includes all or part of an application *[IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering*, 2.1*]* ***6.*** set of grouped change requests, established in the Application Change Management Process, which are designed, developed, tested, and deployed as a cohesive whole *[ISO/IEC 16350-2015 Information technology — Systems and software engineering — Application management*, 4.28*]*

cf. version.

[ISO 24765:2017]

Version… initial release

3.4545 , version

1. **initial release or re-release of a computer software configuration item**, associated with a complete compilation or recompilation of the computer software configuration item *[IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering*, 2.1*]*

2. initial release or complete re-release of a document, as opposed to a revision resulting from issuing change pages to a previous release *[IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering*, 2.1*]*

3. operational software product that differs from similar products in terms of capability, environmental requirements, and configuration

4. identified instance of a configuration item *[ISO/IEC TR 18018:2010 Information technology — Systems and software engineering — Guide for configuration management tool capabilities*, 3.15*]*

5. identified instance of an item *[ISO/IEC 12207:2008 Systems and software engineering — Software life cycle processes*, 4.56*]*

6. unique string of number and letter values indicating a unique revision of an item *[ISO/IEC 19770-5:2015 Information technology — IT asset management — Part 5: Overview and vocabulary*, 3.54*]*

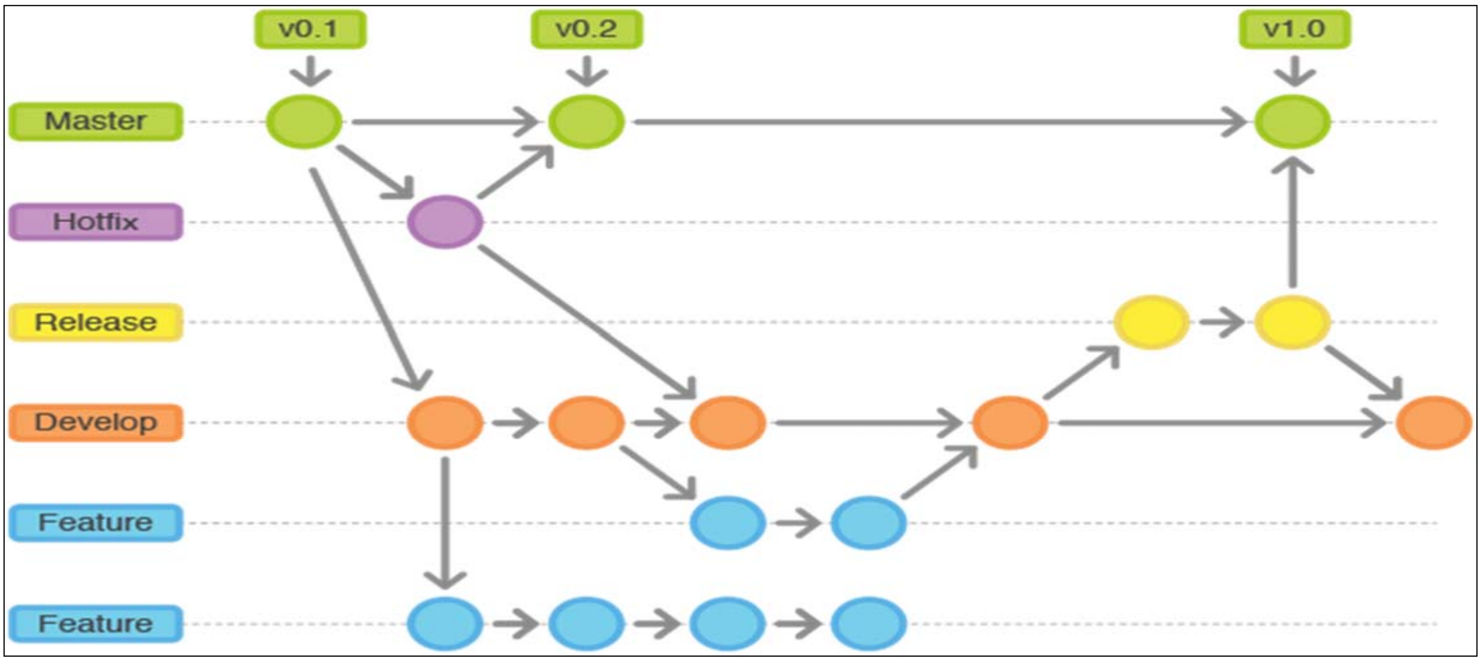
cf. release

Note 1 to entry: Versions often identify revisions of software that provide unique functionality or fixes. A version typically has multiple parts, such as a major version, indicating large changes in functionality or user interface changes, and a minor version, indicating smaller changes in functionality or user interface changes.

[ISO 24765:2017]

Tampereen yliopisto
Tampere University

GitLab versioning, branches



Tampereen yliopisto
Tampere University

Versions

Review at version control tool;

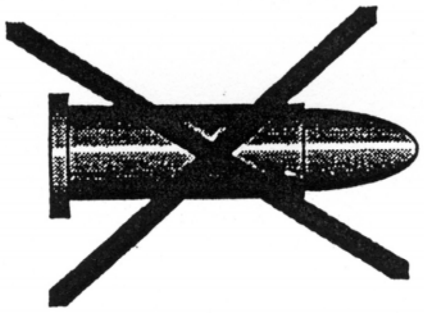
- Merge request (GitLab)
- Pull request (GitHub)

By the way, about version numbers. **What comes after version 1.9 ?**

- 2.0
- 1.10
- 1.9.1
- 1.91
- 1.9b ?

Tampereen yliopisto
Tampere University

There is no silver bullet!

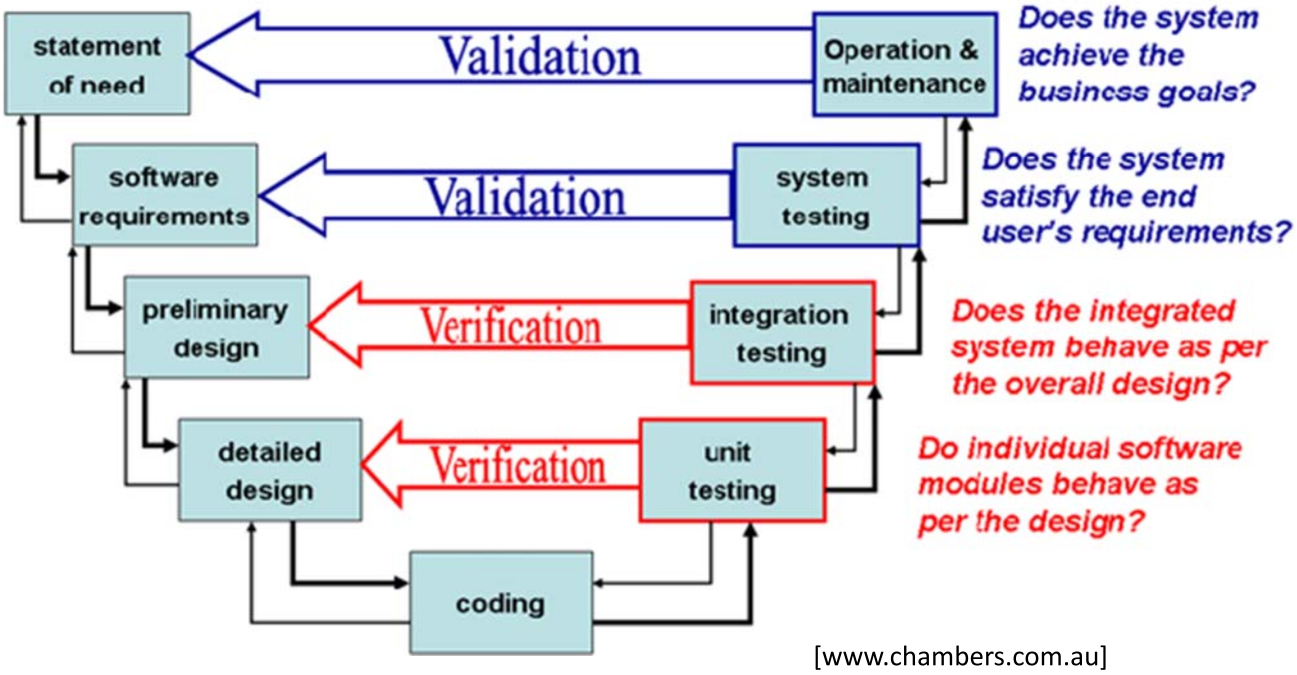


No tool, no method will save you from having to think!

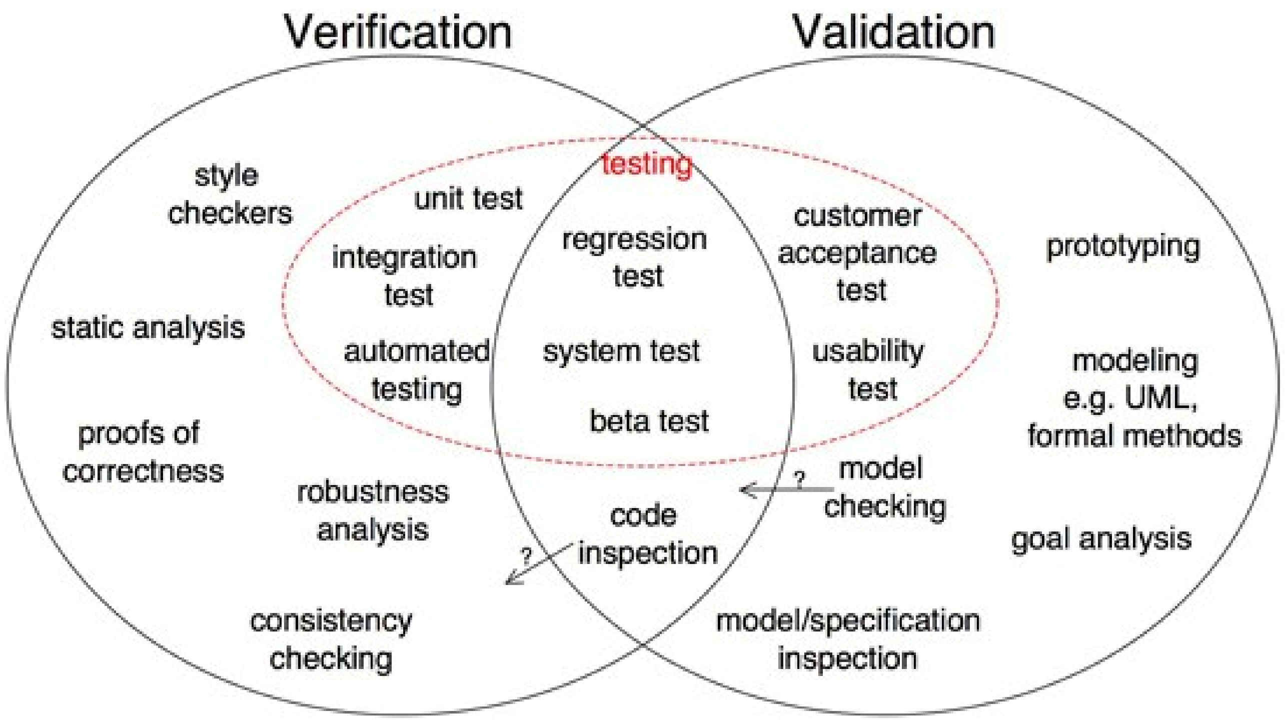
V&V, verification and validation

- **verification**, are we building the product right ?
- **validation**, are we building the right product ?

Dynamic Testing


statement of need
Validation
Operation & maintenance
Does the system achieve the business goals?
software requirements
Validation
system testing
Does the system satisfy the end user's requirements?
preliminary design
Verification
integration testing
Does the integrated system behave as per the overall design?
detailed design
Verification
unit testing
Do individual software modules behave as per the design?
coding

[www.chambers.com.au]



[www.easterbrook.ca]

CM = configuration management

3.779 , configuration management (CM)

1. discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements

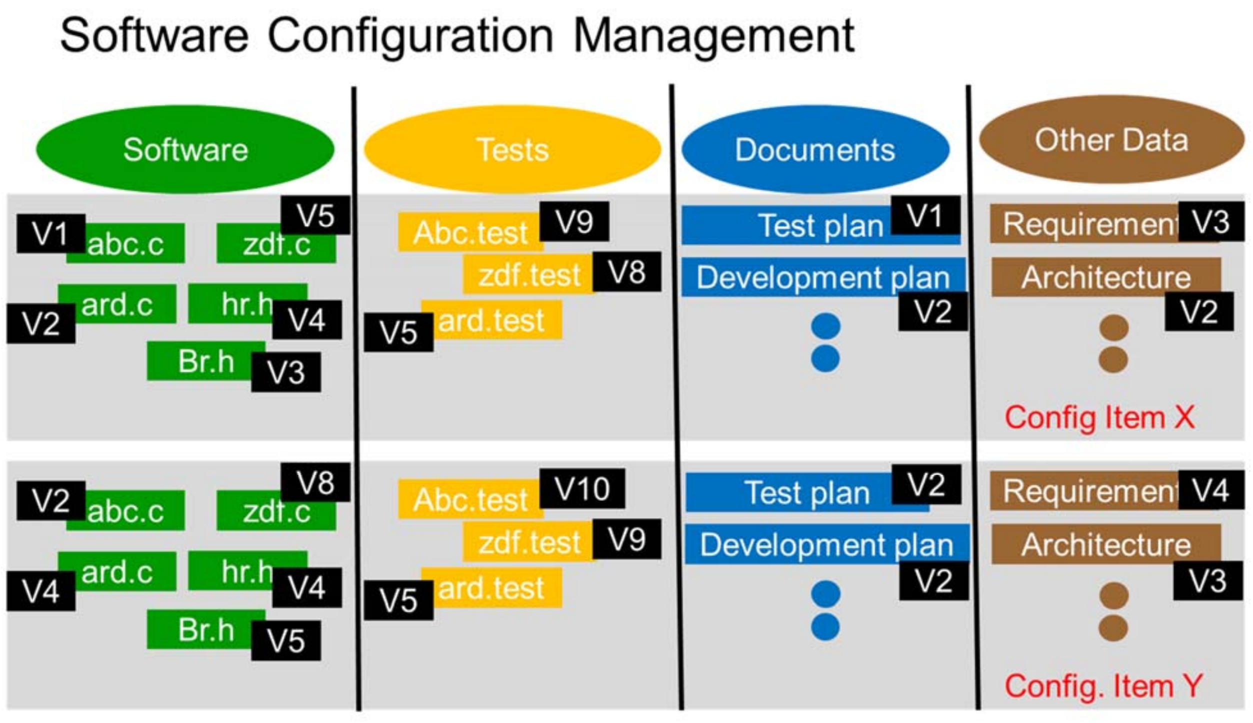
2. technical and organizational activities, comprising configuration identification, control, status accounting and auditing

3. Coordinated activities to direct and control the configuration

cf. baseline, change management, configuration identification, configuration control, configuration status accounting, configuration audit.

[ISO 24765:2017]

You have better to have some good system for managing software configurations



[http://blog.heicon-ulm.de/configuration-management-a-challenging-task/]

Baselines

A *Concept of Operations Baseline*

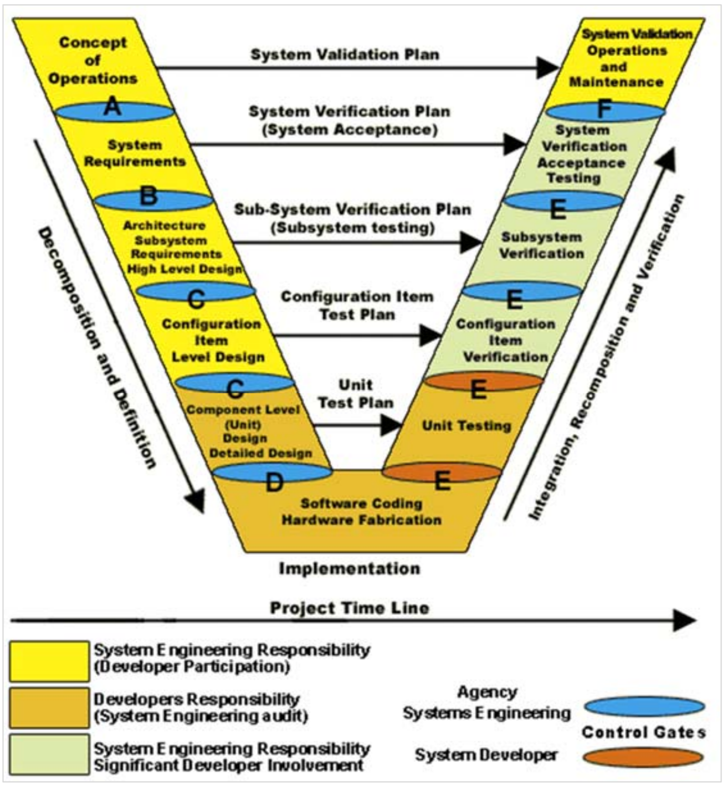
B *System Baseline*

C *Subsystem Baseline*

D *Development Baseline*

E *Product Baseline*

F *Operational Baseline*


Concept of Operations
System Validation Plan
System Validation Operations and Maintenance
A
System Verification Plan (System Acceptance)
F
System Requirements
System Verification Acceptance Testing
B
Sub-System Verification Plan (Subsystem testing)
E
Architecture Subsystem Requirements High Level Design
Subsystem Verification
C
Configuration Item Test Plan
E
Configuration Item Level Design
Configuration Item Verification
C
Unit Test Plan
E
Component Level (Unit) Design Detailed Design
Unit Testing
D
E
Software Coding Hardware Fabrication
Implementation
Decomposition and Definition
Integration, Recomposition and Verification
Project Time Line
System Engineering Responsibility (Developer Participation)
Developers Responsibility (System Engineering audit)
System Engineering Responsibility Significant Developer Involvement
Agency Systems Engineering
System Developer
Control Gates

[https://ops.fhwa.dot.gov/freewaymgmt/publications/cm/handbook/]

Highlights - What to remember

- quality is not something you can add to a product later, it must be in the development process from the beginning
- outsourcing sounds great, but plan and implement it carefully
- software testing is worth automation, e.g. unit tests
- ”ad-hoc” testing without planning is quite useless
- from testing there should be some evidence, too (e.g. test log or diary)
- at a system test you can not test every combination of users’ actions (e.g. at www applications and games), you may use e.g. exploratory testing with some ”story” about what to do
- terminology is not coherent… it varies… depends on adopted process…
- there are many tools to help on e.g. static code analysis, version control and configuration management
-